

BEST AVAILABLE COPY



REC'D 16 JUL 2004

WIPO

PCT

MAGYAR KÖZTÁRSASÁG

ELSŐBBSÉGI TANÚSÍTVÁNY

Ügyszám: P0301368

A Magyar Szabadalmi Hivatal tanúsítja, hogy

AMT Advanced Multimedia Technology AB., Karlshamn (SE),

Magyarországon

2003. 05. 20. napján 19241/03 iktatószám alatt,

Eljárás és berendezés mozgóképadatok tömörítésére

című találmányt jelentett be szabadalmazásra.

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

Az idefűzött másolat a bejelentéssel egyidejűleg benyújtott melléklettel mindenben megegyezik.

Budapest, 2004. év 06. hó 24. napján

Szabó Emilné

A kiadmány hitelül: Szabó Emilné osztályvezető-helyettes

The Hungarian Patent Office certifies in this priority certificate that the said applicant(s) filed a patent application at the specified date under the indicated title, application number and registration number. The attached photocopy is a true copy of specification filed with the application.



Eljárás és berendezés mozgókép adatok tömörítésére

P 0301368

AMT AB, Karlshamn, SE

2003-05-20

Háttér:

Az 1990-es évek elején kidolgoztak egy MPEG1 néven ismert videó kódoló rendszert ami le tudta redukálni a digitális videó adat mennyiséget kb. 1/30 részére. Ez a rendszer minőség problémái miatt tovább lett fejlesztve az MPEG2 néven ismert videó kódolóra, és ezt a kódolót DVD és DVB rendszerekben alkalmazzák elsősorban. Az MPEG4 néven ismert továbbfejlesztett változat elsősorban internet jellegű úgynevezett streaming médiákhoz készült.

E szabadalom célja egy nagy hatékonyságú videó kódoló rendszer javasolt bemutatása ami lehetővé teszi mozgó videó adat tárolását félvezető memóriákban is, és ez által lehetővé teszi a számítógép piacon elterjedt alacsony árú ram memóriák alkalmazását videó tárolására. Ilyen nem mechanikus, nem mozgó tároló rendszer előnyösen alkalmazható televíziós készülékekben, műhold és kábel tv un. settop boxokban a program anyag ideiglenes (átmeneti) tárolására, és a hagyományos szalagos videó magnót kiváltására. A kódoló rendszer szintén előnyösen alkalmazható a videokamerákban lévő mechanikus szalagos felvevő rendszer helyettesítésére a digitális videó adatnak Flash memóriában való tárolásával.

A javasolt **Hibrid Videó Kompresszió** kódoló rendszer lehetővé teszi jó vizuális minőséggel 300-600 kbit/s sávszélességre való csökkentését, ami két órányi videó adatnál nem több mint ca 4-500 Mbyte tárolási kapacitás igényt jelent..

A mozgóképek kódolására több ismert rendszer létezik.

A mozgóképek kódolása során a kódolt adatmennyiség dinamikusan változik a megadott minimális és maximális határértékek között hogy tartható legyen a kívánt minőség illetve a kívánt totális adat hossz.

A program szabályozza a kompressziót annak megfelelően hogy x képből álló szekvencia mekkora átlag hosszt adott.

Amennyiben az átlag hossz nagyobb az előző átlagnál növeli ha kisebb csökkenti a kompressziót a minimum ill. maximum határértéken belül.

A kompresszió növelése erősebb kvantizációt jelent ami nagyobb hibát és zajt okoz. A hiba gyakran nagyon jól látható és zavaró lesz különösen 1Mb/s alatt.

Mivel minden kép komprimálhatósága változó azonos minőség esetén a kívánt átlag hossz nehezen tartható.

A minimum és maximum értékeket nem állíthatjuk túl nagyra mivel a szabályozási tartomány is túl nagy lesz és a kapott teljes hossz is túl széles

skálában változhat.

Gyakori eset hogy a beállított minimum érték nem elegendő a kívánt minőség tartásához és szükségessé válna további csökkentés.

A problémák kiküszöbölése végett a találmány egy megvalósítása szerint bevezetjük a dinamikus skálázás fogalmát.

Amennyiben a beállított szabályzás nem elegendő a minőség tartásához a program leskálázza a képet akkorára ami már kielégítő eredményt ad majd ezt komprimálja, és dekompreszió esetén rekonstruálja az eredeti méretben. Ebben az esetben is van minőség csökkenés de elsősorban a képélességben. Blokkosodás és egyéb tipikus hibák nem jelentkeznek amennyiben a beállított kompresszió nem extrém magas.

Jelölések:

Intra-frame = Frame ami nem használ semmilyen referenciát közvetlenül DCT és kvantizáció után blokkonként van kódolva.

I-Frame = Intra-frame, rövidítése **I**.

P-Frame = Previous frame, rövidítése **P**.

B-Frame = Backward frame, rövidítése **B**.

Current = Aktuális frame (amit kódolunk), rövidítése **C**.

MSE = Measure error.

Inter-blokk = Minden blokk ami az előző vagy az utána következő referencia frame blokkjai segítségével let létrehozva. Az inter-blokk tehát minden olyan blokk ami **P** vagy **B** vagy a kettő lineárisan interpolált átlagából készült.

Az inter-blokk tehát egy általános kifejezés ami **P** vagy **B** eljárást takar és olyan esetekben alkalmazzuk amikor lényegtelen a leírás szempontjából hogy milyen a konkrét típusa.

Amennyiben blokkokról beszélünk az **I,P,B** a kezdő betűk által jelzett frame típushoz tartozó blokkot jelöli. A továbbiakban minden esetben külön jelezzük ha eltérünk ettől a jelölésektől.

Az egyik leginkább elterjedt rendszer a video adatok sűritésére az MPEG.

(1. ábra)

Tekintsük át röviden a működését.

A rendszer az egymás utáni képekből csak a megváltozott részeket kódolja.

Ezt a gyakorlatban úgy valósítja meg hogy a képeket felbontja blokkokra és összehasonlítja az előtte levő ill. az utána következő képekben lévő blokkokkal.

(2. ábra)

Az összehasonlításhoz a következő összefüggést használják:

$$MSE(k,l;u,v) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I_n(k+i,l+j) - I_{n-1}(k+i+u,l+j+v))^2$$

Az előtte és az utána következő képeket nevezzük egységesen referencia képeknek. A keresés során a referencia képen létrehoz egy keresési tartományt -32/32 mérettel. Az aktuális blokk pozíciója lesz a középpont.

Vagyis ha éppen a 10,10 pozíciónál tartunk akkor a referencia képben ez a pozíció lesz a középpontja a tartománynak.

A tartományon végig lépteti az aktuális blokkot és minden lépésnél kiszámítja a hibát. Az a pozíció lesz a legjobb jelölt ahol a legkisebb hibát kaptuk.

A kapott hiba nagysága alapján eldönthető hogy a keresés sikeres vagy sem.

Sikeres keresés esetén megkaptuk teljes képpont felbontásban a kereset pozíciót viszont a kapott eredmény még nem kielégítő.

Ha megvizsgáljuk láthatjuk hogy a mérési módszer az oka a hibának.

Például zajos kép esetén a legjobbnak ítélt pozíción a két blokk nem biztos hogy tökéletesen fedi egymást mivel a zaj miatt a blokkok információ tartalma különböző.

Ahoz hogy a blokkok jól fedjék egymást ki kell szűrni a zavaró tényezőket.

A hibák kiszűrésére aluláteresztő szűrőket alkalmaznak.

A szűrők a beállított minavételezési frekvencia alapján csillapítják a magas frekvenciás komponenseket vagyis elnyomják a zajt.

Gyakorlatilag átlagolnak minden pixelt a mellette vagy alatta lévővel vagy mind a kettővel és ezáltal kompenzálják a hibákat. (ún. 1/2 pixel felbontás)

A hibák kompenzálására bilineáris szűrő javasolt.

Minden referencia képből létrehoznak három interpolált (szűrt) referencia képet.

Horizontálisan interpolált vertikálisan interpolált ill. mindkét irányban interpolált képet.

A következő fázisban 1/2 pixel pontossággal folytatjuk a keresést.

A megtalált x,y pozíció utolsó biteiből létrehoznak egy szelektort ami kiválasztja a kívánt interpolált képet. $S = (y \& 1) * 2 + x \& 1$.

Létrehozunk az első fázis által megtalált pozícióval a középpontban egy új keresési tartományt -1/+1, -1/+1 határértékkel. $(x + (-1/+1), y + (-1/+1))$.

Ebben a tartományban újra indítjuk a keresést.

A szelektor kiválasztja a pozíciók alapján az szükséges interpolált referencia képet és a pozíciók által mutatott blokkot a képben és elvégezi a négyzetes hiba kiszámítását az aktuális blokkhoz képest.

A keresés végén azt a pozíciót tartja meg ami a legkisebb hibát adta.

Az aktuális blokkot tehát abból az interpolált blokkból fogjuk kivonni amire a szelektor mutatott amikor a legkisebb hibát kaptuk. (3. ábra)

A továbbiakban ha a keresés sikeres volt a kapott különbségeket ha pedig sikertelen magát az aktuális blokkot Diszkrét Koszinusz Transzformációval idő reprezentációról frekvencia reprezentációra alakítjuk majd csökkentjük a felesleges precizitását (kvantizáció.)

Az így kapott értékek nagy része nulla lesz ami entrópia kódolással hatékonyan kódolható a kapott pozícióval együtt. (4-5. ábra)

Mivel a dekódolásnál nem áll rendelkezésre az eredeti kép mint referencia így csak egy már dekódolt képet használhatunk annak.

Szükségszerű már a kódolás során a referencia képeket a kódolt képből létrehozni. Ezt úgy érhetjük el legegyszerűbben ha rögtön a kvantizáció után elvégezzük az inverz kvantizációt és inverz DCT.

Amennyiben a keresés sikeres volt az előzőekben hozzáadjuk a megtalált referencia blokkot a vissza transzformálthoz. A vissza transzformált blokkot az aktuális kép aktuális pozíciójába írjuk. Az aktuális kép minden blokkja frissítve lesz mivel az így létrehozott kép lesz a következő kép referenciája.

Sikeres keresés esetén a blokk inter egyébként intra blokknak lesz minősítve.

Ez szükséges a dekódernek hogy tudja milyen módon lett a blokk létrehozva a kódolás során. Ezen az elven vannak a képek is minősítve.

Ha a képnek nincsen referenciája vagy a kép teljesen megváltozott a referenciákhoz képest akkor az egész kép DCT -val van transzformálva és "I-Frame-nek" van minősítve. Amennyiben a kép csak az előző képet használhatja referenciának "P-Frame-nek" van minősítve. Ha pedig mind az előzőt mind az utána következőt használhatja referenciának "B-Frame-nek" van minősítve.

B képtípus esetén mind az előzőben mind az utána következőben keresés azt tartja meg ami a kisebb hibát adta vagy a kettő lineárisan interpolált átlagát

Először a kiszámítja a megtalált P és C blokk MSE értékét majd a megtalált F és C blokk MSE értékét. A következőkben kiszámítja $0.5 * (P + F)$ képlettel előállított blokk és a C MSE értékét. A három alternatíva közül azt a blokkot kódolja ahol a legkisebb MSE értéket kaptuk. Vagyis ha a megelőzőhöz viszonyítva adta a jobb eredményt akkor a P blokk lesz a referencia ha az utána következő adta a jobb eredményt akkor az F blokk ha pedig az F és a P átlaga akkormindkettő. Ha a három alternatíva közül egyik sem teljesül akkor intra blokként lesz kódolva. Minden esetben a blokk leíró

struktúrában jelezni kell hogy honnan származik a referencia vagyis milyen módon lett kódolva. (6. ábra)

A "P" típusnak a sajátossága hogy referenciának csak "I" vagy "P" típust használhat.

A "B" típusnak a sajátossága hogy referenciának csak „P” típust használhat.

Sikeres keresés esetén a kapott pozíciókat vektorokká konvertáljuk és kódoljuk.

A vektorok tehát megadják hogy a blokk a referenciához képest mennyit és milyen irányban mozdult el.

A DCT meglétét a következő tények igazolják.

Ha a koefficiensek egy részét töröljük (nullázzuk) az inverz transzformáció képes igen jó közelítéssel rekonstruálni az eredeti adatokat.

Felmerülhet a kérdés miért DCT alkalmazunk amikor a DCT az FFT -nek egy változata.

A válasz az hogy jobb a függvény közelítése mint az FFT-nek. (7. ábra)

A kvantizáció feladata a felesleges pontosság (részletezettség) csökkentése.

Ha közelebbről megvizsgáljuk a blokkot látjuk hogy a blokkban sok olyan részlet van amit a szemünk nem érzékel mivel az emberi szem érzékenysége az alacsony frekvenciás komponensek felé növekszik.

Ha tehát a magasabb frekvenciás komponenseket erősebben elnyomjuk mint az alacsonyabbakat a képen egy határig nem látható változás viszont a komprimálhatósága növekedett. Ezt a típusú kvantizációt használja az MPEG1-2.

A másik módszer az hogy nem vesszük figyelembe a koefficiensek frekvencia elrendezését minden koefficienst konstans értékkel osztunk. (H26x – MPEG4)

A kvantizációnak a legfontosabb feladata a koefficiensek bitjeinek a csökkentése vagyis minél kevesebb bittel leírni azt.

Minél kevesebb bittel tudjuk a koefficienst leírni annál jobban növeljük a komprimálhatóságát viszont az egész számú osztásból eredő hiba is annál inkább növekszik.

A bitek számának csökkentésére további módszerek lehetségesek.

DC (delta code) predikció ami arra az elvre épül hogy az egymás után következő blokkok 0. pozícióján lévő értékek csak kevéssel térnek el egymástól. Vagyis a bitek száma csökkenthető ha kivonjuk ezeket egymásból. (A 0. pozíción lévő értéket DC-nek még a többi AC-nek neveznek).

AC predikció hasonló a DC-hez csak különböző irányokban kérdezik le a koefficienseket és különböző módszerekkel átlagolják azokat.

Az AC/DC predikciónak több lehetséges megoldása van ezeket itt nem részletezzük.

A predikción itt olyan reverzibilis matematikai összefüggést értünk ami alapvetően átlagolásokra épül valamint az eredeti értéket jó közelítéssel adja vissza.

A hagyományos entrópia kódoló

A videó technikában leggyakrabban az Huffman kódolást alkalmazzák.

A videó technikában alkalmazott Huffmannal kódolással elérhető eredmények a módszer elméleti teljesítményénél rosszabbak, mivel teljesen statikus és a kódolandó adatokhoz hozzárendelt valószínűségek pontatlanok. Az entrópia kódolások hatékonyabbak, ha adaptívak.

Az eljárás alkalmazását egyetlen tény igazolja a nagy sebesség mivel semmi más nem kell csinálnia mint kimásolni a táblázatból a karakterhez tartozó kódot.

A találmány szerinti tömörítési eljárásban az ún. aritmetikai kódolás egy adaptív változatát használjuk.

Hibrid Videó Kompresszió Leírása

A megvalósított kompresszor hierarchikusan felépített dinamikusan változó méretű blokkra épülő eljárás. A keresés során nem csak az előtte lévő ill. az utána következő referencia képet használhatja hanem az azt megelőző ill. az azután következőket is maximum ± 3 szint mélységben.

Magas szintű mozgás kompenzálás valósít meg $\frac{1}{2}$ től $\frac{1}{16}$ pixel felbontásig.

Az entrópia kompresszor neurális vagy diszkrét predikcióra épülő optimalizált aritmetikai kompresszor.

(8. ábra)

Néhány szó a predikció fogalmához.

Predikción olyan reverzibilis matematikai összefüggést értünk ami tipikusan átlagolásokra épül.

Ennek következtében a visszaállításnál csak közelítő pontossággal tudjuk vissza kapni a kívánt értéket. Mondhatjuk úgyis hogy "megjósoljuk" a várt értéket. A gyakorlatban természetesen egy függvényt használunk ami a számításokat az átlagolások alapján elvégzi.

A Intrapredikció alatt olyan reverzibilis matematikai összefüggést értünk ami a két blokk segítségével az aktuális blokk pixel értékeit csökkenti vagy kinullázza.

Két teljesen különböző predikciós rendszerre hivatkozunk a leírásban:

- 1, Az Intra képeknél mint intra predikció valamint
- 2, A neurális entrópia kódolásnál alkalmazott predikció.

Az egész dokumentben minden esetben az 1. predikcióra hivatkozunk kivéve azokat a részeket ahol a neurális entrópia kódolásról van szó.

Intraprediktív kódolás. (I típus)

Az Y blokk mérete 16x16 vagy 16db 4x4-es blokk.

Az UV blokk mérete 4 db 4x4-es blokk. Hasonló a H26L definícióban leírtakkal.

A predikció arra a megfigyelésre épül hogy a szomszédos blokkok térbeli korrelációi felhasználhatók a redukcióra mivel ezeknek azonos tulajdonságaik vannak.

A predikció az aktuális blokk előtt lévő függőleges vagy a felette lévő blokk vízszintes sorával vagy mindkettővel valósul meg. (9. ábra)

A függőleges sort nevezzük B-nek a vízszintest A-nak.

A 16x16 blokk predikciója

A predikciónak négy lehetséges típusa definiált.

1. DC predikció.

Ha létezik A és B akkor

$$S0 = \sum (A_j + B_j + 16) / 32$$

különben ha csak az A létezik

$$S0 = \sum (A_j + 8) / 16$$

különben ha csak az B létezik

$$S0 = \sum (B_j + 8) / 16$$

Különben $S0 = 128$

$DCP(j,i) = IB(j,i) - S0$ ahol $j=0..15, i=0..15$ (IB aktuális blokk, DCP prediktált blokk)

2. Vízszintes predikció

$$DCP(j,i) = IB(j,i) - A(i) \text{ hol } j=0..15, i=0..15$$

3. Függőleges predikció

$$DCP(j,i) = IB(j,i) - B(i) \text{ hol } j=0..15, i=0..15$$

4. Plan predikció

$$v = 5 * ((\sum ((A_{(j+7)} - A_{(j-7)}) * j)) / 4) / 4$$

$$h = 5 * ((\sum ((B_{(j+7)} - B_{(j-7)}) * j)) / 4) / 4$$

$$k = A_{(15)} + B_{(15)}$$

$$T_{(j,i)} = (k + (i-7) * h + (j-7) * v + 16) / 32$$

$$DCP_{(j,i)} = IB_{(j,i)} - T_{(i)} \text{ ahol } j=0..15, i=0..15$$

A 4x4-es blokk predikciója

A predikciónak hat lehetséges típusa definiált.

1. DC predikció.

Ha létezik A és B akkor

$$S0 = \sum (A_j + B_j + 4) / 8$$

különben ha csak az A létezik

$$S0 = \sum (A_j + 2) / 4$$

különben ha csak az B létezik

$$S0 = \sum (B_j + 2) / 4$$

Különben S0 = 128

$$DCP_{(j,i)} = IB_{(j,i)} - S0 \text{ ahol } j=0..3, i=0..3 \text{ (IB aktuális blokk, DCP prediktált blokk)}$$

2. Vízszintes predikció

$$DCP_{(j,i)} = IB_{(j,i)} - A_{(i)} \text{ hol } j=0..3, i=0..3$$

3. Függőleges predikció

$$DCP_{(j,i)} = IB_{(j,i)} - B_{(i)} \text{ hol } j=0..3, i=0..3$$

4. Diagonális vízszintessel és függőlegessel

$$T_{(0,0)} = (B_{(3)} + 2 * B_{(2)} + B_{(1)} + 2) / 4$$

$$T_{(1,0)} = (B_{(2)} + 2 * B_{(1)} + B_{(0)} + 2) / 4$$

$$T_{(2,0)} = (B_{(1)} + 2 * B_{(0)} + A_{(-1)} + 2) / 4$$

$$T_{(3,0)} = (B_{(0)} + 2 * A_{(-1)} + A_{(0)} + 2) / 4$$

$$T_{(4,0)} = (A_{(-1)} + 2 * A_{(0)} + A_{(1)} + 2) / 4$$

$$T_{(5,0)} = (A_{(0)} + 2 * A_{(1)} + A_{(2)} + 2) / 4$$

$$T_{(6,0)} = (A_{(1)} + 2 * A_{(2)} + A_{(3)} + 2) / 4$$

$$DCP_{(j,i)} = IB_{(j,i)} - T_{(j-i+3)} \text{ ahol } j=0..3, i=0..3$$

5. Diagonális függőlegessel

$$T_{(j,i)} = A_{(3)} \text{ ahol } j=0..3, i=0..3$$

$$T_{(0,0)} = (A_{(0)} + A_{(1)}) / 2$$

$$T(1,0) = A(1)$$

$$T(0,1) = T(2,0) = (A(1) + A(2)) / 2$$

$$T(1,1) = T(3,0) = A(2)$$

$$T(0,2) = T(2,1) = (A(2) + A(3)) / 2$$

$$DCP(j,i) = IB(j,i) - T(j,i) \text{ ahol } j=0..3, i=0..3$$

6. Diagonális vízszintessel

$$T(j,i) = B(3) \text{ ahol } j=0..3, i=0..3$$

$$T(0,0) = (B(0) + B(1)) / 2$$

$$T(0,1) = B(1)$$

$$T(1,0) = T(0,2) = (B(1) + B(2)) / 2$$

$$T(1,1) = T(0,3) = B(2)$$

$$T(2,0) = T(1,2) = (B(2) + B(3)) / 2$$

$$DCP(j,i) = IB(j,i) - T(j,i) \text{ ahol } j=0..3, i=0..3$$

Az első lépésben 16x16-os blokkal kezdjük a feldolgozást.

Elvégezzük a predikciót minden módban és minden prediktált blokkot tárolunk.

A blokkokat transzformáljuk kvantizáljuk és az entrópia kódolóval komprimáljuk teszt módban. Teszt módban a tanulás letiltott és a kimenetett nem tároljuk el csak a kapott hosszt.

A blokkot a következőképpen dolgozzuk fel:

DCT -> Q -> IQ -> IDCT -> IP

Kiszámítjuk visszaállított blokk és az eredeti blokk pixel reprezentációjának a négyzetes hiba arányát.

Azt a transzformált blokkot tartjuk meg ahol a legjobb a hiba arány és a legkisebb a hossz.

A megtartott blokk kvantizált koefficienseit ismételten komprimáljuk de most már normál módban. A következőkben elvégezzük a predikciót és a transzformációkat az UV blokkal is miután az Y blokkot kódoltuk.

Az UV predikciója annyiban tér el az előzőktől hogy csak a DC predikció definiált.

Amennyiben a kapott legjobb hiba arány nagyobb mint a megengedett legnagyobb hiba érték akkor töröljük az előző eredményeket az UV kivételével és megismételjük az eljárást 4x4-es blokk mérettel.

A helyes dekódoláshoz szükséges a predikció típusának a rögzítése.

Továbbá ha a kvantizáció után a blokk összes eleme nulla értelmetlen a kódolás elegendő egy bittel jelezni azt (CBP).

Minden visszaállított blokkot beírunk az új referencia képbe miután megszürtük anti-blokk szűréssel.

Az anti-blokk szűrésnek a feladata hogy megszüntesse a blokkosodást vagyis a blokkról blokkra történő átmenet ne legyen érzékelhető.

Nézzük a fejléc leírását:

A 16x16 blokk kezdetén kódoljuk a blokk típusát 0 vagy 1.

Nulla esetén a blokk 16x16-os mérettel lett kódolva ellenkező esetben 16db 4x4-es részblokkonként lett kódolva.

Következő CBP és a prediktor típusának a kódolása

A 16x16 blokk esetén $CBPPT = CBP * 8 + PREDTYPE$ különben

4x4 blokk esetén $CBPPT = CBP * 4 + PREDTYPE$

A CBPPT minden blokkhoz vagy részblokkhoz kiküldjük.

UV blokkok esetén csak a CBP kódoljuk.

A változások keresése

Mint az előzőekben jeleztük az eljárás VBSM technikára épül.

Az általunk használt lehetséges legnagyobb blokk mérete 64x64.

Ez többek között azt is jelenti hogy a kép méretének 64-el oszthatónak kell lenni.

Amennyiben ez nem áll fent a kép széleihez hozzá kell adni annyi képpontot hogy a feltétel igaz legyen.

Mit kapunk ezzel a technikával. Mindenek előtt a mozgás vektorok számának a csökkenését.

Ha pl. a híradót nézzük ideális eseten többnyire csak a bemondó arca változik.

Egy 768x576-os kép esetén 16x16-os blokkal 1728 vektort kell kódolni.

Még 64x64-es blokkal ~118 vagyis az 1/15 része.

Ez természetesen extrém eset, normális esetre nézve 40-60% -os csökkenésre számíthatunk. (10. ábra)

A 64x64 természetesen a választható legnagyobb blokk méret.

▶ A rendszer folyamatosan átlagolja a legnagyobb blokkal történő keresés hiba arányait és amennyiben egy megadott idő alatt a hiba túl nagy csökkent a blokk méreteit a következő képtől kezdődően.

A legnagyobb blokk semmi esetre sem lesz 16x16-nál kisebb.

Ennek elsősorban a keresésre ráfordított idő szempontjából van jelentősége.

Némileg javítja a minőséget is de nem számottevően mivel a 64x64-es blokkat ebben az esetben mindenképpen felbontotta volna kisebbekre.

A blokkok "quad-tree" struktúrával vannak leíva.

Minden 64x64-es blokk egy gyökér ami négy levélre leszármazott blokkra bontható és minden levél tovább bontható négy újabb leszármazottra egészen 4x4-es blokk méretig. (11. ábra)

A gyakorlatban a legnagyobb blokkal kezdődik a keresés.

A kereső algoritmus EZPS eljárásra épül.

Nem célunk a EZPS ismertetése mivel ez ismert eljárás.

Gyakorlatilag bármilyen kereső eljárás alkalmazható amennyiben az eljárás MSE kiszámítása a későbbiekben leírt modellre épül.

Az MSE értékeket 4x4-es blokkonként képezi és a végén ezeket az MSE értékeket összegzi. Az így kapott MSE összeg az egész blokk hiba leírója.

Hiba térképnek nevezzük a 4x4-es blokkok MSE értékeit amit az egész keresés végéig sorrendben egy külön blokkban tárol.

Bemutatunk két eljárást ami csak a referenciák feldolgozásában különbözik

Az keresést az ún. "full-search" eljárással ismertetjük ami pontról pontra megegyezik a standard MPEG-nél leírtakkal azzal a különbséggel hogy a keresés során egy lépésben hiba térképet is készít.

1. módszer

1. rész (a blokkok keresése)

Kijelöljük a keresési tartományt a referencia képben és megkeressük azt a pozíciót ahol a legkisebb hibát kapjuk.

A hiba meghatározására a következő összefüggést használjuk.

$$MSE(k, l: u, v) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I_n(k+i, l+j) - I_{n-1}(k+i+u, l+j+v))^2$$

A hiba számítása során a teljes blokkot 4x4-es régiókra bontottuk. (12. ábra)

Minden régiónak külön számítjuk a hiba faktorát majd a végén összegezzük azokat.

Az így kapott összegzett hiba szám a teljes blokkra vonatkozik.

A keresés során minden esetben amikor találunk egy kisebb hibát mint az eddigi legkisebb az összegzett hibát az összes régió hiba összegével együtt elmentjük.

2. rész (A megtalált legjobb blokk pozíció további feldolgozása)

Ha a keresés végén az egész blokkra vonatkozó összegzett hiba nagyobb mint a megengedett a fa struktúrája alapján a következő szintre lépünk 4 részre osztjuk a blokkot és összegezzük a régióhoz tartozó hiba faktorokat.

Ismételten összehasonlítjuk a hibákat a megengedettel majd szükség esetén tovább bontjuk a régiókat.

Miután a teljes blokkot feldolgoztuk rendelkezésünkre áll a teljes blokk hiba térképe a fa struktúrája alapján.

Felolvassuk a teljes fát és vissza propagáljuk a hibákat a magasabb régiók felé.

Vagyis a kapott hibák és az aktuális régióhoz tartozó hiba maximum deviációjának segítségével meghatározzuk azt a legmagasabb régiót ahol a keresést a legérdekesebb megismételni.

A rendszer folyamatosan ellenőrzi hogy van -e értelme a visszalépésnek vagy sem.

Ha például csak két hibás 4x4-es blokkunk van a két régióval való visszalépésnél nincs értelme a visszalépést elvégezni mivel nincs arányban a ráfordított munka a nyereséggel.

Ilyen esetekben csak a hibás 4x4-es régiókon ismételjük meg a keresést.

A feldolgozás befejezésekor mindazokat a régiókat amelyeknek az MSE értéke nem kielégítő tovább keressük a következő referencia képben.

A keresés a következő alternatívák közül valósul meg.

1. Keresés csak az előző P képtípusokban.

2. Keresés csak az előző B képtípusokban.

3. Keresés az előző és az utána következő mozgás kompenzált referenciákban.

B típus nem használható ebben az esetben referenciaként.

Az első és a harmadik esetben esetben kombinálhatja a blokkokat egymással.

A keresés mindig a legközelebbi P referenciával kezdődik ha nem ad kielégítő eredményt a távolabbi P referenciával folytatja majd ha ez sem kielégítő akkor kombinálja a két referencia legjobb jelöltjeit egymással.

Ha így sem kielégítő tovább lép a 2. ill. 3. alternatívára.

(13. ábra)

Amennyiben a keresés végén egyik referenciával sem kapunk kielégítő eredményt megjelöljük a blokkot mint intra blokkot és az intra frame-nél leírtak szerint kódoljuk.

Minden esetben ha egy referencia kép I típusból származik az összes előtte lévő referencia törölve lett. Vagyis a keresés visszafelé csak eddig a képig tarthat.

Ez szükséges mivel az I típus csak akkor keletkezik ha a képező teljesen megváltozott az előzőekhez képest.

Vagyis eleve nem lennének használhatóak a pillanatnyi képhez képest mivel a tartalmuk teljesen más.

2. módszer

1. rész: megegyezik az 1. módszer 1. részével.

2. rész

A keresés végén kiértékeljük a kapott négyzetes összeget és ha benne van az elfogatható tartományba aminek a határértékei nagyon szigorúak akkor a keresést befejeztük.

Ellenkező esetben folytatjuk a keresést a többi referenciában is.

Ha egyik referencia sem adott a nagy blokk mérettel kielégítő eredményt akkor azzal a referenciával folytatjuk ahol a legkisebb négyzetes összeget kaptuk.

A blokkot felbontjuk 4 régióra majd a régiókhoz tartozó hiba térképből származó négyzetes hibákat összegezzük.

Ezek alapján eldöntjük hogy melyik régiók a hibásak.

Ezeket a fent leírt módon tovább bontjuk addig amíg az összes blokk elfogadhatóvá nem válik vagy elértük a legkisebb blokkot.

Ha tehát a hiba továbbra is fenn áll akkor vissza propagálja a hibát a egy vagy két lépéssel a magassabb régiók felé vagyis megkeresi azt a nagyobb blokkot ami tartalmazza a hibát.

Ezt a régiót összehasonlítja a többi referencia kép azonos régióival.

Ha valamelyik referencia képen a régió elfogadható akkor ezt felhasználja.

Ellenkező esetben elvégzi a keresést a hibás régióval.

Ha a keresés után a régió továbbra is hibás akkor visszamegy arra a referencia képre ahol a legkisebb méretű hibás régiók a legkisebb hibát mutatják és a program megvizsgálja a hibás régiók elhelyezkedését. Ha a hibás régiók egymás alatt vagy egymás mellett helyezkednek el összevonja a hibás régiókat és megismétli a keresést az összevont régiókkal.

Amennyiben a hiba még mindig fenáll és a státusz információk azt mutatják hogy a lineáris interpoláció megengedett interpolálja a hibás blokkokat a két legjobb referencia kép blokkjaival.

Elenkező esetben kódolja a blokkot intra prediktív módon az I képtípusnál leírtak szerint.

Mint leírtuk a keresés EZPS eljárásra épül de ugyanígy alkalmazható más gyors kereső eljárás is "Diamond" stb.

Mind a két módszer végén a régiók méretének az optimalizálása következik.

Összevonja azokat a régiókat ahol men volt szükség a keresés megismétlésére.

A bemutatott rendszer hatékonyan lecsökkenti a felesleges keresések és mozgás vektorok számát.

A második módszernek nagyobb a számítási igénye mivel teljes blokk mérettel egyszer el kell végeznie a keresést a referencia képekben.

A referencia képek felhasználásának a szabályai azonosak az első módszerrel.

1. és 2. módszer 3. rész: mozgás kompenzálás

Minden megtalált blokkhoz adaptáljuk a mozgás kompenzálást vagyis elvégezzük a keresést $\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$ és $\frac{1}{16}$ -od pixel felbontásban a blokkok méretei és a beállított precizitás alapján. Másszóval ha a blokk mérete 16×16 , $\frac{1}{4}$ pixel lesz a legnagyobb

precizitás amennyiben ez megengedett. Nagyobb blokkok esetén csökken a precizitás még kisebb blokkok esetén növekedik.

A kompenzált blokkokat kivonjuk az eredeti blokkokból és egy átmeneti 64x64-es blokkba írjuk ugyanazokra a pozíciókra ahogy az eredetiben is szerepel.

Miután az eredeti 64x64-es blokk minden részét feldolgoztuk prediktáljuk az átmeneti blokkot a fa struktúrája alapján az intra predikciónál leírtak szerint azzal a különbséggel hogy a predikció 8x8-as méretre is definiált.

Minden prediktált blokkot DCT transzformációval transzformálunk kvantizálunk majd entrópia kódolással kódolunk továbbá a kódoló kimenő kódjait nem tartjuk meg csak a kód hosszát írjuk le.

A kódoló teszt módban működik ami azt jelenti hogy a tanulás letiltott nem módosíthatja az adatbázisát.

Ez nyilvánvalóan hibát okoz de arra jó hogy meghatározzuk hogy melyik predikció adja a legrövidebb kódot.

Minden blokk ami 16x16-nál nagyobb 16x16-os részblokkra van bontva és minden részblokk eszerint van prediktálva és kódolva.

Minden intra blokk kivan zárva a predikcióból tehát a szomszédos blokkok predikciója során nem használhatja fel az intra blokk adatait.

A tesztelésnek az első lépése az hogy predikció nélkül kódoljuk majd predikcióval.

Miután elvégeztük a blokk tesztelését azt a módot tartjuk meg ahol a legkisebb hibát és a legrövidebb kódot kaptuk.

Ismételten elvégezzük az entrópia kódolást a kiválasztott mód transzformáltjain de most már normál módban tanulóval.

A DCT DC értékeit kivonjuk egymásból de csak az azonos méretű blokkból.

A fent leírt eljárás igen hatékony de legrosszabb esetben ha a 64x64-es blokkban nagyon sok a 4x4-es részblokk és nagyon sok lesz a predikció típusát leíró adat is.

Ennek csökkentésére kidolgoztunk egy alternatív eljárást is.

Miután az eredeti 64x64-es blokk minden részét feldolgoztuk az átmeneti blokkot 16x16-os blokkokra bontjuk majd elvégezzük a predikciókat a blokkokra.

Ha valamelyik 16x16-os blokk intra blokkot tartalmaz azt kizárjuk a 16x16-os predikcióból és 4x4-es méretű predikcióval dolgozzuk fel a benne lévő nem intra blokkokat.

A továbbiakban az átmeneti blokkot az eredeti részblokk méreteivel transzformáljuk és kódoljuk.

A leírt eljárás mindkét alternatívája elsősorban P képtípusokra vonatkozik.

B képtípus esetén annyiban módosul hogy a mozgás kompenzálásnak ki kell terjedni az UV szín felületekre is, továbbá mivel az előtte és az utána következő képekben lévő blokkok átlagát is képezheti az átlagolásnak a hiba képzését meg kell előznie ugyanis az átlagolt blokk és az eredeti blokk különbségéből származtatjuk a hibát.

A felhasználható referenciák száma.

Elvileg nincsen akadálya a keresésnek még több referencia képben viszont sokkal nagyobb processzor teljesítményt igényelne.

Ha megvizsgáljuk látjuk hogy egy processzor aminek 1 órajel ciklus szükséges minden utasítás feldolgozásához CIF(352x288) felbontásban 16x16-os blokkokkal számolva 640Mhz-es órajelre lenne szüksége a keresés megvalósításához.

A modern processzorok képesek 16 instrukciót egyidejűleg feldolgozni így 41Mhz elegendő volna. B képtípus esetén a duplájára lenne szükség hiszen kettő referencia képen kell keresni ha pedig 4 referencia képünk van akkor a négyszerese.

Ez 160Mhz CIF felbontásban még teljes felbontásban ennek a négyszerese.

A mozgás kompenzálás megvalósítása

A kompenzálás a pillanatnyi megvalósításban $\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$ $\frac{1}{16}$ pel felbontású a blokk méretétől függően.

A tesztek azt mutatják hogy nincs értelme 8x8-as blokk méret felett $\frac{1}{4}$ pel felbontásnál nagyobb felbontást alkalmazni ugyanis a megtalált blokk csak megközelíti az eredetit. Más szóval nem a mozgás precizításában hanem a blokk információ tartalmában van a különbség.

Ilyen esetben nincs értelme a precízebb kompenzálásnak ugyanis nem ad jobb eredményt.

A kompenzálás a gyakorlatban a következőképpen valósul meg.

Az 1/1 felbontásban megtalált legjobb pozíció körül létrehozunk 10 tesztponthoz spirális elrendezésben.

Az eredeti blokkot kivonjuk a tesztponthoz kezdődő n lépésben minta vételezett referencia blokkból. (n=2,4,8,16)

A kapott különbségeket alul-áteresztő szűrővel megsűrjűk.

A szűrő tipikusan 6-8 lépcsőből áll a típusát nem részletezzük mivel a konkrét megvalósítása fejlesztés alatt áll.(bevált típusok bikubik, Hadamman és/vagy wiener).

A wiener szűrővel kicsivel jobb képélesség tapasztalható.)

A szűrt blokknak leképezzük a hibáját és abszolút értékben összegezzük azokat.

A hiba leképezésén az eredeti és a szűrt blokk pixeljeinek különbségét kell érteni.

A kapott különbségeket abszolút értékben összegezzük ami a hiba mutató lesz.

(minél kisebb az értéke annál kisebb a hiba.)

Miután mind a 10 pozíciót leteszteltük az a pozíció lesz a legjobb ahol a hiba a legkisebb.

A különböző kompenzálások felbontás szerint sorba vannak kapcsolva.

Amennyiben a a legnagyobb felbontás $1/8$ akkor először elvégezzük $1/2$ pixel felbontásban a keresést majd a kapott pozíciótól kezdődően az $1/4$ -es keresést és az így kapott pozíciótól kezdődően az $1/8$ -ost.

A három alternatíva közül ismét a legkisebb hibát választjuk.

(14. ábra)

Az alkalmazott entrópia kódolás

Az aritmetikai eljárás

Az aritmetikai kódolás valós számra épülő eljárás.

A valós szám tartománya 0 és 1 közé esik.

Ezt a tartományt osztjuk fel úgy résztartományokra hogy minden karakter kapjon egy saját tartományt aminek a nagysága az előfordulási gyakoriság szorzata.

PL: az EMMA szó a következőképpen nézne ki:

Karakter gyakoriság tartomány

A $1/4$ 0.0 - 0.25

E $1/4$ 0.25 - 0.5

M $2/4$ 0.5 - 1.0

Az első karakter kódolása annyiból áll hogy a tartománya értékeit betöltjük az L és a H változóba. Miután az első karaktert kódoltuk tudjuk hogy a kimenet 0.25 és 0.5 közé esik. Minden további új karakter módosítani fogja ezt a tartományt.

Minden új tartományt a következő formulával számolunk:

$$\text{DIFF} = H - L$$

$$L = L + \text{DIFF} * l$$

$$H = L + \text{DIFF} * h$$

Ahol l és a h az új karakterhez tartozó tartomány alsó és felső értéke.

A következő karakter amit kódolunk az M

L: $0.25 + (0.5 - 0.25) * 0.5 = 0.375$ és

H: $0.25 + (0.5 - 0.25) * 1 = 0.5$ lesz.

A következő ismét M

L: $0.375 + (0.5 - 0.375) * 0.5 = 0.4375$ és

H: $0.375 + (0.5 - 0.375) * 1 = 0.5$ lesz.

És az utolsó az A

L: $0.4375 + (0.5 - 0.4375) * 0.0 = 0.4375$ és

H: $0.4375 + (0.5 - 0.4375) * 0.25 = 0.453125$ lesz.

Tehát a kódolt üzenet 0.4375 lesz amiből a teljes üzenet rekonstruálható.

A dekódoláshoz szükség van a frekvencia táblára hogy a tartományokat meghatározhassuk. Az első karakter dekódolásához megvizsgáljuk hogy a kapott 0.4375 melyik tartományba esik. Jelen esetben a 0.4375 a 0.25 – 0.5 tartományba esik amihez az E karakter van hozzárendelve.

Következőkben kivonjuk a tartomány alsó felét a 0.4375-ből és elosztjuk a tartomány felső és alsó különbségével. A kapott eredmény 0.75 lesz ami a 0.5-1.0 közé esik és a hozzárendelt karakter M.

Megismételjük az eljárást a 0.75 ami 0.5 fog adni vagyis ismét M lesz.

Ismételjük az eljárást a 0.5-el ami 0.0-át fog adni vagyis a hozzá tartozó tartomány a 0.0– 0.25 amihez az A van rendelve.

A fent leírt eljárás az alapja az aritmetikai kódolásnak.

A gyakorlatban azonban ez nem praktikus mivel valós számokat használ.

Megvalósítható a probléma egész számokkal is de figyelembe kell venni hogy az egészszámmal való ábrázolás korlátozott precizitást jelent.

Tehát előfog fordulni alul és túl csordulás amit a programnak le kell tudni kezelni.

Másodszor nem engedhetjük meg hogy első lépésben a frekvenciát analizálja majd utána kódoljon mivel nem áll rendelkezésünkre minden adat.

Ebből adódóan csak adaptív megvalósítás jöhet számításba.

Megvizsgáltuk hogy milyen megvalósítás lenne az adott problémához a legcélszerűbb és eredményül a bináris aritmetikát kaptuk.

Minden beérkező karaktert felbontunk bitekre és bitenként kódoljuk.

Nézzük hogyan valósul meg egész számokkal.

Létrehozunk két külön frekvencia táblát a nulláknak és az egyeseknek aminek minden elemét 1-el töltünk fel. (N,E) (N=nullák táblázata,E=egyesek táblázata)

p=prediktor

$L = 0$ $H = 2^{24}$ $i = 0$ (tábla index) $n = 0$ (bit számláló) C az aktuális karakter

ismétlés ha ($n < 8$)

Ha $(C \text{ and } 128) = 128$ akkor bit = 1 különben bit = 0

$i = i * 2 + \text{bit}$

$p = N(i) / (N(i) + E(i))$

$K = L + (H - L - 1) * p$

Ha $K = L$ akkor $K = K + 1$

Ha bit = 1 akkor $E(i) = E(i) + 1$ és $L = K$

Ha bit = 0 akkor $N(i) = N(i) + 1$ és $H = K$

ismétlés ha ($H - L < 256$)

$H = H - 1$

Kíírás $(L / 65536)$ eredményének az alsó byte-ja.

$L = L * 256$

$H = H * 256$

Ha $L > H$ akkor $H = 2^{24}$

ismétlés vége

$n = n - 1$

$C = C * 2$

Ha $n = 0$ akkor $i = 0$

ismétlés vége

A fent bemutatott eljárás egyenértékű a Huffmannel viszont annál jóval egyszerűbb.

A kódolás hatásfoka a (p) predikciótól függ.

Minél hatékonyabb a predikció annál jobb kompressziós effektivitást kapunk.

Ha a predikciót dinamikus Markov modell alapján készítenénk el nagyon jó eredményt kapnánk viszont a prediktornak a meghatározásához az előző bit kombinációkat és a hozzájuk tartozó frekvenciákat hierarchikusan tárolni kellene ami óriási memória igényt jelentene.

Gyakorlatilag nem lenne felső határa ha nem limitálnánk.

Limitálva az optimális érték kb: 16Mb ahol jó effektivitással működne.

Jó lenne meghatározni a következő bit valószínűségét és a prediktort ennek figyelembe vételével kiszámítani.

Ebben az esetben ugyanis nem egy bitet vizsgálunk hanem egy bit sorozatot.

Ha pl: a fent ismertetett eljárással kódolnánk egy 100Kb hosszú blokkot amiben periodikusan A és B karakter ismétlődik 13Kb kapnánk.

Ha viszont a prediktor a bitek viszonyát is vizsgálja akkor 4-6 Byte-ot.

A dinamikus Markov modell képes erre az analízisre ennek köszönhető a nagy memória igény.

A probléma ugyanis az hogy egy C állapotba eljuthatunk A -ból is ill. B-ből is.

A C-nek nincsen információja hogy honnan jutottunk ide így amikor a következő bit beérkezik a D vagy az E -be jutottunk lehetetlen lesz a korrelációk meghatározása.

A megoldás az hogy a C-t klónozni kell.

Ebben az esetben az A-ból csak a C-be juthatunk majd onnan a D vagy az E-be ill.

A B-ből csak a C'-be és onnan a D vagy az E-be.

Így már meghatározhatók az összefüggések, mondhatjuk úgyis hogy a hálózat lépésről lépésre megtanulta az összefüggéseket.

Ez adta az alap ötletet az általunk megvalósított eljáráshoz.

A beérkező bitet egy n bit hosszúságú léptető regiszterbe töltjük.

Az n értéke attól függ hogy milyen hosszú szekvencia korrelációit akarjuk vizsgálni.

Tipikusan 32 bit.

A teljes léptető regiszter értékéből létrehozunk m számú indexet.

Más szóval a kijelölt memória területet m részre osztjuk úgy hogy a területek az első 256 byte kivételével átlapolják egymást és egymással uniókat képeznek.

Az unióknak helyzetük és nagyságrendjük alapján kiemelt jelentősége van.

A memóriának minden tartománya kap egy szorzó faktort ami 1 ha az adott tartomány nem unió.

Az uniók fontossági sorrendben vannak állítva nagyságuk és valószínűségük szerint és ezekből a faktorokból származik a konstans értékük is.

Minden index rámutat egy memória pozícióra amiben a nullák és az egyesek előfordulási gyakorisága szerepel.

A gyakoriságok és a prioritási konstansok segítségével kiszámítjuk az új prediktor értékét.

A következő beérkező bit és a prediktor alapján leképezzük a hiba faktort majd módosítjuk az indexek által mutatott gyakoriságokat a hiba a prioritás és a beérkezett bit alapján.

Mindezek után léptetjük be a bitet a regiszterben és határozzuk meg az új prediktor értékét a fent leírtak alapján.

A felvázolt megoldás diszkrét logikára és számításra épül. Az aritmetikai kódolás konkrét megvalósítását részletesen ismertetjük a leírás végén.

A másik megoldás a neurális hálózat technikájából ismert **preceptronra** épül.

A hálózat három rétegből áll. Bemenő réteg közbenső réteg és kimenő réteg.

Megoldható kettő réteggel is viszont lényegesen pontatlanabb eredményt ad.

Minden az indexek által címzett memória pozíció a neurális hálózat egy bemenetét írja le. Tartalmazza az egyesek és a nullák gyakoriságát továbbá az adott bemenet aktivitási állapotát.

A predikció kiszámításánál a neurális hálózat összegzi az aktivitási állapotokat majd egy normalizált szigmoid függvényen keresztül kiszámítja a prediktort.

A diszkrét megoldásnál leírtuk a hiba képzést amit itt is alkalmazunk.

Módosítjuk az indexek által mutatott gyakoriságokat a beérkezett bit alapján.

A hiba a prioritás és a beérkezett bit alapján módosítjuk az aktivitási állapotokat.
(tanulási fázis)

A gyakorlati tesztek során megfigyeltük hogy pontosabbá tehetjük a predikciót ha előtanítjuk a hálózatot.

Mivel a kódolandó adat struktúrák képről képre hasonlóak meghatározható egy optimális kezdő adatbázis.

Gyakorlatilag egy képet addig tanítottunk amíg a hiba faktor kisebb nem lett mint az előre beállított optimálisnak tartott érték.

Sok képen megismételtük az eljárást majd a kapott adatbázisok átlagából létrehoztunk egy kezdő adatbázist.

A kódoló adatbázisa 0.5Mb ami abból adódik hogy fel van bontva funkciók szerint.

Külön kódoljuk a DCT koefficienseket külön a mozgás vektorokat és külön minden egyéb működés szerint szétválasztható adatot.

Összesen 7 szint van és minden szinthez tartozik egy 64Kb előtanított adatbázis.

Egyedül a koefficienseket kódolásánál használunk 2 szintet.

Ha a koefficiens nulla a 7. szinten egy nulla bitet kódolunk egyébként egy egyest és a hatodik szinten kódoljuk magát a koefficienst.

Ez a megoldás kevesebb bittel írja le a blokkot mint ha csak egy szinten kódoltuk volna.

Továbbá ideálisabb mintha a nullákra run-length kódolást alkalmaztunk volna

Az kódoló teljes egészben egészszámú aritmetikára épül beleértve a neurális hálózatot is és optimalizált VCL ill. MMX logikára.

Az átviteli sebessége 3.89Mb/s.

A kompressziós hatásfoka lényegesen jobb mint a hagyományos UVLC és a H26x-ben alkalmazott CABAC-nak. (15. ábra)

Az átviteli sávszélesség és a kompresszió szabályozása

Vizsgáljuk meg hogy miért és milyen szabályozásra van szükség.

A mozgóképekben a képek információ tartalma széles skálán változik.

Ez maga után vonja a kompresszió változását is széles skálán amennyiben állandó képminőséggel számolunk.

Amennyiben a komprimált adatokat valamilyen médian szeretnénk tárolni a média kapacitása behatárolja a lehetséges adat mennyiséget.

Ha pedig valamilyen rendszeren továbbítjuk az adatokat valós időben a rendelkezésünkre álló sávszélesség (másodpercenként mennyi adatot tudunk továbbítani) szab határt.

Tehát szükséges hogy a másodpercenként adatmennyiséget közel állandó szinten tartsuk. Ezt csak a kompresszió szabályozásával érhetjük el.

Szabályozás alatt a kvantizáció növelését vagy csökkentését kell érteni.

Ez azt vonja maga után hogy ha növeljük a kvantizációt egyre több részlet tűnik el a képből egyre alacsonyabbak lesznek a térfrekvenciák és egyre inkább növekedik a látható hiba. Egy bizonyos ponton túl a kép szétesik kockákra és egyéb torzulásokra.

A hagyományos megoldások kiszámítják a várt hossz és a kapott hossz alapján az új kvantizációs faktor és ezt alkalmazzák a következő képmezőre.

A finomabb megoldások a megadott időegység alatt komprimált képmezők átlaghosszából és a várt átlaghosszból képezi az új kvantizációt.

Ezek a megoldások beiktatnak egy reakció várakoztatási faktort is ami megadja hogy mennyi idő után kell a szabályozásnak elérnie a a kiszámított maximumot.

Ezek a megoldások konstans átviteli sebességet adnak. (CBR constant bit rate)

Sokkal jobb eredményt érhetünk el ha az adott szabályozás figyelembe vesz egy minimum és egy maximum határértéket és az átvitelt e két határ közé szorítja lehetőleg úgy hogy hosszú távon a kettő átlag középértéke legyen a domináns.

Továbbá figyelembe veszi az eredeti és a rekonstruált kép közötti jel/zaj viszonyt és ennek figyelembevételével végzi a szabályozást.

Vagyis ha a jel/zaj viszony romlott növeli egyébként csökkenti az átviteli sebességet a megadott határokon belül. (VBR variable bit rate)

Ennek a megoldásnak a hátránya hogy pontosan megjósolhatatlan a várt totális hossz. A minimum és maximum értékeket nem állíthatjuk túl nagyra mivel a szabályozási tartomány is túl nagy lesz és a kapott teljes hossz is túl széles

skálában változhat. Gyakori eset hogy a beállított maximum érték nem elegendő a kívánt minőség tartásához és szükségessé válna további növelés.

A megvalósított megoldás

A megvalósított modell backpropagációra épül ami a perceptron tovább fejlesztett általánosított változata kibővítve egy Kohonen réteggel ami osztályozást végez.

A Kohonen rétegnek a feladata eldönteni hogy a backpropagáció által előrejelzett értékek milyen kvantizációnak ill. skálázó faktornak felelnek meg.

A hálónak a feladata megjósolni a következő képmező ideális kvantizációs faktorát.

A bemenetein megkapja a képre kiszámított hosszt és a kapott hosszt valamint a képminőséget és a deviációs faktort.

A kimeneten megkapjuk a K (kvantizációs) és S (skálázó) faktort.

A K értékét felhasználja a következő azonos típusú képnél még a S értékét átlagolja a következő "I" típusú képig. (lásd a dinamikus skálázásnál)

A neurális háló előre tanított rendelkezik a teljes tudás alappal.

Ez azt jelenti hogy képről képre beállítottuk az ideális K és S értéket az összes paraméter figyelembevételével és ezt tételesen megtanítottuk a hálózatnak.

Természetesen a tanuláshoz szükséges adatok előkészítése sokkal bonyolultabb mivel hatalmas mennyiségű adatról van szó ezért különböző transzformációkkal korreláció számításokkal kezelhető szintre hoztuk.

Minden képtípusokhoz (I,B,P) hozzá van rendelve egy előre tanított adatbázis ami a hálózat aktivizációs állapotait tartalmazza.

Mondhatjuk úgyis hogy külön hálózat létezik minden képtípusra valójában azonban egy hálózat három előtanított adatbázissal ami a kép típusa alapján választódik ki.

Ez szükséges mivel nem azonos a kvantizáció nagyságrendje a különböző képtípusokra.

(16. ábra)

A neurális hálózatnak több választható tudás bázisa van amikből különböző üzemmódokat állíthatunk be.

CBR

VBR

2 lépcsős CBR

2 lépcsős VBR

Minden tudás bázisból két implementáció létezik egy ami dinamikus skálázással készül és ami annélkül.

A CBR módnál a hálózat nem veszi figyelembe a minimum és maximum határértékeket csak a jel/zaj viszonyt amennyiben a skálázás megengedett.

Két lépcsős módban az első lépcsőnél nem használ semmilyen szabályozást konstans

kvantizációs faktorról kódolja a teljes adatfolyamot továbbá a képet nem skálázza.

A második lépcsőnél a kvantizációs faktort nem a következő képmezőre számítja hanem az éppen feldolgozás alatt lévőre továbbá nem végzi újra a keresést mivel az első lépésben már elvégezte és a pozíciókat leírta az adatfolyamban.

Tehát csak az DCT és a kvantizáció valamint a kvantizált blokkok kódolása történik újra a második lépésben.

A fent leírtakkal lényegesen jobb vizuális minőség érhető el mint a hagyományos módszerekkel mivel az emberi szem vizuális igényét követi az elérhető legjobb kompresszióval.

A hagyományos módszereknél a minőség és a határfok gyakran kizárja egymást.

(17. ábra)

A dinamikus skálázás

A előzőekben leírtuk hogy gyakran előfordul olyan eset amikor a kódolás nem tudja a kívánt hosszt tartani az előírt minőségben vagy a beállított kompresszió extrém nagy bizonyos szekvenciákhoz képest és a megadott minimum és maximum határértékeken belül nem lehet a kívánt minőséget tartani.

Jó példa erre a Múmia 2 című film első 5 perce.

Ehhez a szekvenciához > 2.5Mbit/sec beállítás kellene ha jó minőséget szeretnénk kapni egy jó MPG kompresszióval.

Ha lemegyünk 1.5Mbit/s -ra mind a kompresszió mind a dekompresszióval nagyon komoly elő és utó szűréseket kellene végezni a hibák kiszűréséhez ami a képélességet jelentősen rontaná. Az így kapott képet csak az „elfogatható” kategóriába lehetne sorolni.

Megvizsgáltuk hogy mi történne ha ezeknél a szekvenciáknál a bemenő adat mennyiséget csökkentenénk de a beállított kompressziós faktorokat változatlanul hagynánk.

Vagyis ha a képet leskáláznánk kisebbre $\frac{3}{4}$, $\frac{1}{2}$... részére.

Mivel a kódoló eljárás arra törekszik hogy az adatokat lehetőleg állandó szinten tartsa a minőség figyelembe vételével a méret csökkenése a kompresszió

csökkentését fogja maga után vonni. Így ugyanazt az adat mennyiséget fogjuk kapni mint a skálázás előtt a kimeneten.

Ha tehát 0.5Mbit/s-al kódoljuk a teljes adatfolyamot és a kritikus részeknél a képméretet a felére csökkentettük akkor ezeknél a részeknél a tényleges átviteli sebesség továbbra is 0.5Mbit/s lesz viszont a kvantizációs faktor és a kompresszió az n -ed részére csökkent ami olyan képminőségnek felel meg mintha eredetileg 2Mbit/s -al kódoltunk volna.

Ez maga után vonja a hibák lecsökkenését.

A módszer hibájául elmondható hogy a skálázással csökkentjük a kép felbontását.

Vagyis a dekódoló oldalon miközben a képet rekonstruáljuk az eredeti méretében csak következtetni tudunk a hiányzó képpontok értékére.

Ezt nagyban javítani tudjuk megfelelő skálázó eljárás alkalmazásával ha kiindulunk abból hogy a kép frekvenciából áll és a transzformációt ennek a figyelembe vételével végezzük.

Több interpoláción alapuló skálázó eljárást teszteltünk és a legjobb eredményt a Láncos módszere adta.

Ha összehasonlítjuk a kompressziót a kritikus részeknél szűréssel és anélkül 0.5Mbit/s -os átviteli sebesség mellett azt kapjuk, hogy a kritikus részeknél skálázás nélkül a veszteség nagyon jól látható.

A kép erősen blokkosodik szálkásodik sok helyen a teljesen ellaposodott és a képélesség drasztikusan lecsökkent. (minta kiradírozták volna)

A találmány szerint végzett skálázással mindezek a hibák nem jelentkeztek. Az egyetlen érzékelhető hiba a képélesség csökkenése.

Kielemezve a szekvenciák tartalmát ahol a skálázás szükséges elmondhatjuk hogy tipikusan olyan helyeken szükséges ahol a mozgás gyors. A gyors mozgások esetén a kép már eleve életlenebb mint egyébként így vizuálisan a veszteség alig érzékelhető.

Gyakorlati megvalósítása

Minden bejövő kép keresztül halad a skálázó rendszeren először 0-ás faktorial. **(nincs skálázás)** A kompressziót szabályozó rendszer a fent leírtak alapján megvizsgálja hogy a kódolás kielégítő eredményt ad a megadott határon belül vagy sem. Ha igen nincs változás továbbra is az aktuális faktorial folytatja a kódolást a következő „I” képmezőig. A következő „I” képmezőnél az előző „I” képmezőtől számított átlag S értéke lesz az új skálázó faktorial.

A méret változtatás mindig az I képmezőnél történik az összes utána következő képmező az I méreteit örökli. (18. ábra)

Kvantizáció

Két típust különböztetünk meg.

Non-uniform:

$$F'[u, v] = \text{round} (F[u, v] / q[u, v]).$$

(MPEG)

és uniform.

$$F'[u, v] = F[u, v] / N$$

Ahol minden értéket ugyanazzal a konstansal osztunk.

(H263)

Az alkalmazott kvantizáció a beállított üzemmód függvénye.

Üzemmódok:

1. H263
2. MPG
3. H263/MPG

Ha a kódolás során a 3. üzemmód a beállított 900Kbit/s határt felett automatikusan MPEG kvantizációt használ amennyiben a szekvenciák átlag PNSR értéke nagyobb mint 33dB egyébként H263 -at.

A rendszer működésének összefoglalása

A kompresszió során a képek keresztül haladnak a skálázó rendszeren.

A rendszer eldönti hogy a képet milyen módszerrel kódolja tovább. (I,P,B)

I képező esetén a minden blokkhoz elvégzi a különböző predikciókat és kiválasztja azt ahol a legjobb mérési eredményt kapta majd DCT-vel transzformálja kvantizálja és a megfelelő szinten komprimálja.

P képező esetén az előző képet még B képező esetén az előzőeket és az utána következő képeket használja referenciaként és ebben keresi az aktuális blokkot.

A megtalált blokkot a blokk méretétől és a beállított precíziótól függően kompenzálja majd prediktálja transzformálja és kódolja.

A kapott pozíciókat vektorokká alakítja kivonja az előzőkből majd a megfelelő szinten komprimálja. A szabályozó rendszer a várt és a kapott hossz valamint a minőség figyelembevételével szabályozza a kompressziót. Amennyiben a minőséget nem

lehet tartani a beállított határokon belül lecsökkenti a kép méretét akkorára ahol az eredmény kielégítővé válik. A kép méretét soha nem csökkentheti a felénél kisebbre.

Kiértékelés:

A megvalósított eljárás a vártnál sokkal jobb lett.

Gyakorlatilag 450kB/s határig alig van kifogásolni való vizuális hiba a skálázásból adódó élesség csökkenését leszámítva a kritikus részeknél.

Elmondhatjuk hogy 380-450kB/s sebesség mellett egy átlagos videó készülék SP módjának minőségét hozza még 280-380 között egy átlagos videó készülék LP módjának a minőségét. Ha pedig a sebesség >500kB/s egyre közelebb kerül a DVD minőségéhez. Gyakorlatilag 750kB/s határ felett szemmel már nagyon nehéz összehasonlítani a DVD-vel.

Hátrányául elmondható hogy amennyiben léptetni szeretnénk a képet előre vagy hátra(tekercselés) csak előre meghatározott idő egységgel lehetséges tipikusan >10sec. Ez a probléma az entrópia kompressziótól származik ugyanis az adatbázisát minden adathoz frissíti és ha több képet ugrunk előre az adott képnél az adatbázis hibás lenne mivel nem tudnánk hogy hogyan változtatta volna a hiányzó képek alatt. Ezt a problémát csak úgy tudjuk kiküszöbölni hogy az adatbázist egyenletes időközönként alaphelyzetbe állítjuk. A megadott időközök szabályozzák meg a minimális lépés közt. A megadott idő túl rövid nem lehet mivel rontaná a kompresszió hatásfokát. A legjobb kompressziót természetesen az időközök elhagyásával kapnánk de természetesen nem lehetne ebben az esetben "tekercselni". A minimálisan megadható idő 1sec az optimális 10sec. Kimerevíteni ill. lassítani a képet probléma nélkül lehet.

A neurális aritmetikai kódolóról elmondható hogy normál módban nagyon gyors még teszt módban extrém gyors.

Hibrid Videó Dekoder működése (21. ábra)

A kép rekonstrukciója az bemenő puffert(121) feltöltésével és az információs blokk dekódolásával kezdődik(133).

Az információs blokk (133) tartalmazza a kép eredeti méretét a dekódoláshoz szükséges előtanított adatbázisok típusának kódjait valamint egyéb olyan adatokat ami az egész dekódolásra állandó. Az információs blokk feltöltésére csak egyszer a dekódolás elején kerül sor. A következő lépés a kép fejléc (122) információinak a dekódolása.

A kép fejléc információs blokk a kép pillanatnyi méretét a kép típusát (I,B,P) a használt kvantizátor típusát valamint egyéb csak erre a képre jellemző adatokat tartalmaz.

Amennyiben a kép típusa intra dekódoljuk a blokkok információit és a DCT koefficienseket (126) elvégezzük az inverz transzformációt (127,128,129) minden blokkon és a kapott vissza transzformált blokkokat az új kép memória területére írjuk (131).

Az intra képben minden blokk tartalmazza a rekonstrukciójához szükséges adatokat. (A predikció típusát, valamint hogy lett a blokk felbontva 16x16-os blokként vagy 4db 4x4-es blokként stb.)

Inter képtípus esetén először a fa struktúrájának a dekódolására kerül sor (123).

A fa struktúrája tartalmazza a rekonstrukcióhoz szükséges adatokat.

A fa struktúrája alapján dekódoljuk a régiókhoz tartozó DCT koefficienseket mozgás vektorokat és predikciók kódjait valamint a kódolás során használt a referencia képeket azonosító kódokat.

Elvégezzük az inverz transzformációkat (127,128,129) majd a referencia kép (125) mozgás vektorok által kijelölt blokkjait (124) hozzáadjuk (130) a vissza transzformált blokkokhoz.

Amennyiben valahol lineáris interpolációval lett kódolva először a referencia képek (125) és mozgás vektorok által kijelölt blokk (124) alapján előállítjuk az interpolált blokkot majd ezt adjuk hozzá a vissza transzformált blokkhoz. Minden visszaállított blokkot az új kép memória területére (131) írunk.

Mind az intra mind az inter által visszaállított képet beírjuk a referencia kép (125) területre.

A referencia képterület (125) több képből áll hogy mennyiből az annak a függvénye hogy kódolás során mi volt a legtávolabbi kép amiből a referenciát vette.

A referencia képterület cirkuláris, minden új kép beírásakor a legrégebbi törlődik.

A következő lépés a kép méretének az eredetire történő visszaállítása (132).

A visszaállítás Láncos eljárással történik. Mind a kódolásnál mind a dekódolásnál a skálázást egy szubrutin valósítja meg abban az esetben ha ez szükséges.

Ha olyan hardware videó forrásunk ill. kimenetünk van ami képes a skálázásra a kódoló ill. dekódoló csak a kép méreteit adja meg.

Neurális dekóder

A neurális aritmetikai dekóder megegyezik a fent leírt általános elvvel.

Mivel a módszer adaptív először dekódolunk egy bitet majd a bit segítségével kiszámítjuk az új prediktort.

A prediktor kiszámításánál a kódolásnál alkalmazott neurális hálózatot változtatás nélkül alkalmazzuk.

Vagyis a kóder/dekóder között a különbség egyedül a standard aritmetikai számításban nyilvánul meg a többi teljesen azonos.

A konkrét megvalósításban három sor csak a különbség.

Egy komplett videókódoló / transzkódoló rendszer leírása. (19. ábra)

A videókódoló rendszer alkalmas analóg videó jelek digitalizálására és effektív kódolására valamint tárolására. A rendszer ugyanakkor lehetővé teszi digitális már kódolt videó adat átkódolását, ami effektívebb tárolást eredményez. Ilyen transzkódolás alkalmazható DVB (digital video broadcast) MPEG transzport csomagjainak kb. 20 Mbit/s sávszélességről 600 Kbit/ sávszélességre történő csökkentésére, például műholdvevő és televíziós programok tárolása, vagy digitális kamerák bejátszása mechanika nélküli tárolására.

A kódoló rendszer bemenetei az analóg videó bemenet (93), a komplett összegezett MPEG dekódolt digitális videó és audio csomag bemenet (94), valamint a (105) analóg audió bemenet.

A kódolónak következő üzemmódjai vannak.

a, Az analóg videó jel (93) és az analóg audio jel (105) digitális konverzió utáni kódolása.

b, A digitális videó jel és a digitális jelcsomaggól (94) a demultiplexel (109) leválasztott audio jel transzkódolása.

A szelektor (96) által kiválasztott digitális videóadat (97) a (98) kódolóval kódolva lesz. A kódolt (99) videó adat a digitális audio adattal összetett csomaggá (101) lesz multiplexálva (100) multiplexer segítségével. A (102) digitális csomag a PSC Periferial System Controller (102) segítségével a (103) merevlemezre, optikai médiára, vagy a (104) félvezető memóriába tárolva lesz. A Szelektor (107) által kiválasztott digitális audio jel a (108) kódoló segítségével kódolva lesz, majd az előzőekben leírtak alapján tárolásra kerül.

A tárolt videó és audio adat dekódolása (fig. 20)

A (104) valamint a (105)- ben tárolt adatcsomag a demultiplex (110) segítségével kódolt digitális videó adatra (111) kódolt digitális audio adatra (112) van szétválasztva. A (111) videó adat a (113) dekóder segítségével dekódolva lesz. A dekódolt (114) videó adat szűrés és skálázás (115) után egy digitál/analóg átalakító segítségével (116) analóg videó jellé (117) lesz alakítva. A kiválasztott digitális audio

adat (112) a dekóder (118) segítségével dekódolva lesz. A dekódolt audio adat a digitál/analóg konverter (119) segítségével analóg audio jellé lesz alakítva.

A továbbiakban az ábrákon látható elemeket ismertetjük.

1. ábra Hagyományos MPG4 blokkséma.

1. Bejövő digitális videó adat
2. Intra/inter szelektor
3. Diszkrét Koszinusz Transzformáció
4. Kvantizátor
5. Inverz kvantizátor
6. Inverz diszkrét koszinusz transzformáció
7. A referencia képhez tarozó Intra/inter szelektor
8. Referencia kép puffer
9. Mozgás kompenzáláshoz tartozó filter 1,2,3
10. A szűrőt kiválasztó kapcsoló
11. Változások keresése
12. Minta kódolás
13. Koefficiensek kódolása
14. Videó csatorna kiválasztása
15. Kimenő kódolt adat

2. ábra A blokkokra bontott kép egy blokkjának a keresése az előző ill. az utána következő képben.

16. Megelőző kép.
17. Aktuális kép (aminek blokkjait keressük az előző ill. utána következő képben.)
18. Következő kép.
19. Az utána következő kép keresési tartománya
20. A keresendő blokk
21. Az előző kép keresési tartománya.

3. ábra Keresés és mozgás kompenzálás.

22. Aktuális kép részlete.

- 23. A referencia képen kijelölt keresési tartomány.
- 24. Aktuális blokk
- 25. A megtalált blokk (legjobb egyezés)
- 26. Aktuális blokk tartalma
- 27. A megtalált blokk x,y pozíciójából képzett 2 bites szelektor.
- 28. A megtalált blokk tartalma.
- 29. Az aktuális blokk és a pillanatnyi referencia blokk hiba arányának kiszámítása vagy a két blokk kivonása.
- 30. A kapott négyzetes hiba abszolút értékben véve vagy a megtalált blokk és az aktuális blokk pixeljeinek különbsége.

4. ábra Az intra kép kódolásának a blokksémája.

- 31. Bemenő YUV2 formátumú kép
- 32. Diszkrét Koszinusz Transzformáció
- 33. A felesleges precizitás csökkentése (kvantizáció)
- 34. Delta pulzus kód moduláció
- 35. Run length kódolás.
- 36. Statikus Huffman entrópia kódolás kódolt kimenettel
- 37. A ZigZag 8x8-as méretű blokk frekvenciák alapján történő lekérdezése.(ZigZag)

5. ábra A megtalált és kompenzált P típusú blokk kódolása.

- 38. Az aktuális kép blokkjai.
- 39. A referencia kép keresési tartománya.
- 40. Aktuális kép kódolandó blokkja.
- 41. A referencia képben a legjobb egyezés.
- 42-43 A kereset és a megtalált blokk hiba képzése.
- 44-45 A képzett hibák luminance és crominance komponensei.
- 46. Transzformáció kvantizáció és run-length kódolás.
- 47. A run-length kódok entrópia kódolása.

6. ábra A és B képtípus működése

- 48. Előző referencia kép keresési tartománya.
- 49. Aktuális kép keresési tartománya.

50. Utána következő kép keresési tartománya.

7. ábra A DCT és FFT összehasonlítása

- 51. bemenő adat
- 52. FFT koefficiens
- 53. DCT koefficiens
- 54. vágott FFT koefficiens
- 55. vágott DCT koefficiens
- 56. IFFT-vel rekonstruált adat
- 57. DCT-vel rekonstruált adat
- 58. IFFT-vel rekonstruált adat ábrázolása
- 59. DCT-vel rekonstruált adat ábrázolása

8. ábra A hibrid videó kódoló blokksémája

- 60. Bemenő videó adat.
- 61. Képskálázó modul.
- 62. Kódolást szabályozó modul (neurális kompresszió szabályozás, képváltozás stb...)
- 63. Intra inter kapcsoló.
- 64. Diszkrét koszinusz transzformáció.
- 65. Kvantizáció
- 66. Inverz kvantizáció
- 67. Inverz diszkrét koszinusz transzformáció
- 68. Mozgás kompenzáció. (beállítható $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ pixel felbontásban.)
- 69. Anti-blokk szűrés. (feladata a blokkosodás megszüntetése)
- 70. Az aktuális referencia kép. (A blokkjai automatikusan frissítődnek.)
- 71. A változások keresése Quad-Tree struktúrával.
- 72. Neurális aritmetikai kompresszió.
- 73. A kódolt videó adat.
- 74. Kompenzált mozgás adat

9. ábra A predikció bemutatása

- 75. A prediktált blokk
- 76. A baloldali blokk pixeljei (Vertikális prediktor)
- 77. A felette lévő blokk pixeljei (Horizontális prediktor)
- 78. A lekérdezés irányai (vertikális,horizontális,diagonális)

A lehetséges predikciók száma 5.

- 1. DC predikció
- 2. Horizontális
- 3. Vertikális
- 4. Diagonális
- 5. Horizontális vertikális és diagonális

10. ábra A hagyományos blokk felbontás összehasonlítása a quad-tree felbontással.

- 79. A kép hagyományos 16x16 blokkokkal való felbontása.
- 80. A kép quad-treevel való felbontása. Mint látható a 80. részábrán egy blokk szükség esetén további részblokkokra van bontva.

11. ábra A quad-tree bemutatása

- 81. A blokk részekre történő felbontása.
- 82. A blokkok fa struktúrával való bemutatása.

12. ábra A mérési hibáinak régiókra történő felosztása.

- 83. A megtalált blokk tartalmazza a mérési hibákat.
- 84. A mérési hibák első felbontása.
- 85. A mérési hibák további felbontása.

13. ábra Az alkalmazott keresési üzemmódok bemutatása B képtípusokra

A keresés a következő alternatívák közül valósul meg

- 13a. A. Keresés csak az-előző két P képtípusban.
- 13b. Keresés csak az előző két B képtípusban.
- 13c. Keresés az előző és az utána következő mozgás kompenzált referenciákban.
B típus nem használható ebben az esetben referenciaként.
- 14. ábra A mozgás kompenzálásnak a bemutatása.

- 14a. A megtalált pozíció teljes pixel felbontásban.
- 14b. A helyes pozíció.

16. **ábra** A dinamikus skálázás karakterisztikájának a bemutatása.
A skálázás az "A" pontnál kezdődik és a "B" pontnál ér véget.

- 86. A szabályozás kezdete.
- 87. A szabályozás vége.
- 88. A átviteli karakterisztika dinamikus skálázással.
- 89. A átviteli karakterisztika dinamikus skálázás nélkül.

16. **ábra** A neurális szabályozás blokksémája

17. **ábra** Az entrópia kódoló karakterisztikájának a különböző üzemmódokban egymással és a CABAC-al történő összehasonlítása.

- 90. Átviteli görbe előre tanított adatbázissal a szabadalomban leírt neurális kompresszióval.
- 91. Átviteli görbe tanítás nélkül a szabadalomban leírt neurális kompresszióval.
- 92. A H26x CABAC kompresszor átviteli görbéje.

18. **ábra** A dinamikus skálázás vizuális bemutatása.

19. **ábra** Digitális videó bejátszó rendszer blokksémája

- 121. Bemenő analóg videó jel
- 122. Összetett digitális video és audio jel csomag
- 123. Videó analóg/digitál átalakító
- 124. Analóg és digitális videó bemenő jel kiválasztása
- 125. Kódolandó digitális videó adat
- 126. A szabadalom videó jel kódolója a 8. ábra az előzőleg leírtak szerint
- 121. Kódolt / transzkódolt digitális videó adat
- 122. Digitális kódolt audio valamint a digitális kódolt videó jel multiplexálása
- 123. Tárolásra kész összetett digitális audio-videó adat
- 124. Periferial storage controller
- 125. Mágneses/optikai tároló egység

- 126. Félvezetű (SDRAM,FLASH) memória tároló egység
- 127. Bemenő analóg audio jel
- 128. Analóg / digitál átalakító
- 129. Az analóg csatornáról bejött és a digitális csatornáról bejött digitális audio adat kiválasztása.
- 130. A kiválasztott audiocsatorna valamely standard eljárással (Mpeg,AC3,MP3,etc való kódolása (tömörítés)
- 131. Bejövő összetett digitális audio / videó csomag szétválasztása videó és audio adatcsomagra.

20. ábra Digitális videó lejátszó rendszer blokk-sémája

- 121. Periferial storage controller
- 122. Mágneses/optikai tároló egység
- 123. Félvezetű (SDRAM,FLASH) memoria tároló egység
- 124. A tárolt összetett digitális audio / video csomag szétválasztása videó és audio adattá
- 125. Kódolt digitális videó adat
- 126. Kódolt digitális audio adat
- 127. A szabadalmi leírásban leírt kódoló rendszer által kódolt digitális videó adat dekódolása
- 128. Dekódolt digitális videó adat
- 129. Digitális szűrő és skálázás
- 130. Digitális dekódolt videó adat analóg jellé való átalakítása
- 131. Analóg videó jel
- 132. Digitál audio dekóder
- 133. Digitális audio adat analóg jellé való átalakítása
- 134. Analóg audio jel

21. ábra A szabadalmi leírás által kódolt videó adat dekódolásának blokk-sémája

- 121. A bemenő adat puffer.
- 122. A kép információs blokkjának dekódolása.
- 123. Quad-tree és mozgás vektorok dekódolása.
- 124. A referencia képből a mozgás vektor által mutatott blokk kikópiázása.
- 125. A referencia kép.
- 126. A DCT koefficienseinek dekódolása.

- 127. A koefficiensek inverz kvantizációja.
- 128. Inverz diszkrét koszinusz transzformáció.
- 129. Inverz predikció.
- 130. Az eredeti blokk rekonstruálása a referencia és a dekódolt hiba blokk alapján.
- 131. Az új kép memóriája.
- 132. A kép méretének az eredeti méretre történő nagyítása szükség esetén.

Bináris többszintes aritmetikai kódoló leírása Videó kompresszióhoz

Az entrópia kódoló bináris adaptív technikára épül, ami annyit jelent, hogy a beérkező adatokat bitenként dolgozza fel az eddig beérkezett bitek gyakorisága, valamint mintája alapján.

A beérkező bit bekerül egy n bit hosszúságú léptető regiszterbe.
A regiszter kezdőértéke nulla.

$$reg = 2 * reg + bit$$

A regiszter értéke valamint a bitszámláló alapján létrehozunk egy indexet, ami a frekvencia tábla egy elemére mutat.

$$index = bitcnt * 2^N + reg, \text{ ahol az } N \text{ a regiszter hossza tipikusan } 8 \text{ és } bitcnt=0..N-1$$

A bitszámláló minden beérkező bitnél, egyel növekedik egészen addig amíg nem éri el az N értékét ahol lenullázódik.

A frekvencia tábla tehát $N=8$ -al számolva 2048 bejegyzést tartalmaz.

A frekvencia táblázat minden bejegyzéséhez két frekvencia változó van rendelve egy a nulláknak és egy az egyeseknek. (f_0, f_1)

Az aritmetikai eljárás néhány globális változót használ a tartomány felbontásához
 $low = 0, mid=0, high=(2^{32})-1, p=0.5, bitcnt=0, index=0, reg=0$ kezdőértékkel.

A kódolás menete a következő egy bitre:

$$mid = low + (high - low) * p$$

$$\text{If}(bit = 1) low = mid$$

$$\text{If}(bit = 0) high = mid$$

$$FreqTbl[index].f_0 = FreqTbl[index].f_0 + (2 * bit)$$

$$FreqTbl[index].f_1 = FreqTbl[index].f_1 + (2 - (2 * bit))$$

$$reg = 2 * reg + bit$$

$$Index = bitcnt * 2^N + reg$$

$$bitcnt = bitcnt + 1$$

$$\text{if}(bitcnt = N) bitcnt = 0$$

$$p = \frac{FreqTbl[index].f_0}{FreqTbl[index].f_0 + FreqTbl[index].f_1}$$

PREDIKCIÓ

```

If((high - low) < 256)
{
  outdata =  $\frac{high}{2^{24}}$ 
  low = low * 256
  high = high * 256 + 255
}

```

A kódolt adat kiírása.

A fent bemutatott eljárás az alapja az általunk használt megoldásnak. Ha megvizsgáljuk az eljárást láthatjuk, hogy tartalmaz néhány nem praktikus elemet. Az első mindjárt, az hogy a középérték kiszámításakor valós számot használ. (p) Valós szám alkalmazása nem célszerű ugyanis sokkal több gépi ciklusba kerül az elvégzésük.

Helyette egész számú aritmetikát alkalmaztunk, ahol a p értéke 0 és 65535 közé esik.

Szükségessé vált a középérték számítását módosítani mivel a középérték maximuma 2^{32} még a p érték maximuma 2^{16} így ennek a két értéknek a szorzata túlcsoordulást okozhat mivel az összes változó 32 bites.

A középérték számítását a nagyságrend alapján résztartományokra bontjuk.

$diff = high - low$

if($diff > 2^{28}$)

$mid = mid + \frac{diff}{2^{16}} * p$

else if($diff > 2^{24}$)

$mid = (mid + \frac{diff}{2^{12}} * p) / 16$

▪
▪

A másik problémát a frekvenciák osztása okozza mivel az osztás a legtöbb gépi ciklust igénylő művelet.

Tipikusan 316 ciklust igényel egy osztás még egy szorzás csak 50-et. Így egy frakció tábla alkalmazásával kicseréltük az osztást szorzásra.

$$FracTbl_{(I)} = \frac{2^{26}}{I} + 0.5 \quad \text{ahol } I = 0..512$$

$sum = FreqTbl[index].f0 + FreqTbl[index].f1$

$$p = \frac{FreqTbl[index].f0 * FracTbl[sum]}{2^{10}}$$

Mivel a frakció tábla 512 elemet tartalmaz garantálni, kell hogy a „sum” ami a tábla indexe lesz, ne lépje át ezt a határértéket. Ezt úgy érjük el hogy megvizsgáljuk a „sum” értékét és újra skálázzuk a frekvenciákat.

if (sum > 256)

$$FreqTbl[index].f0 = \frac{FreqTbl[index].f0}{2}$$

$$FreqTbl[index].f1 = \frac{FreqTbl[index].f1}{2} + 1$$

Ezekkel a változtatásokkal kiváltottuk az összes osztást, valamint elértük hogy egy bit kódolásához csak 2 szorzás szükséges.

Ne zavarjon meg senkit, hogy az egyenletekben továbbra is szerepelnek osztások. Ha megfigyeljük, az osztok mindenhol a 2-nek a hatványai, így ha az osztandót jobbra léptetjük annyiszor ahányszor az osztó hatványkitevője mutatja, ugyanazt az eredményt kapjuk, mintha az osztást hagyományosan elvégeztük volna. Értelemszerűen, ha balra léptetünk, akkor szorzunk. A változók léptetése csak néhány gépi ciklust, igényel.

A kódoló továbbfejlesztett változatában, elértük hogy egyáltalán nincs szorzás és osztás, továbbá teljesen kompatibilis a fent leírtakkal.

A kódoló felbontása szintekre

Több szint alatt azt kell érteni, hogy minden szintnek van egy saját frekvencia táblája, aminek a hossza az adott adat típusára van beállítva, valamint inicializálva.

A szintekre való felbontás analóg azzal az elvvel mintha N darab külön kódolót alkalmaznánk egy közös kimenettel.

Ha például lenne 2 adatstruktúránk, amit külön kódolnánk és kapnánk az egyikre 1.5 bpc (bit/karakter) míg a másikra 2.5 bpc-t akkor, ha ezt egy kódolóval kódolnánk a kapott eredmény nagy valószínűséggel a kettő összegénél rosszabb lenne.

A gyakorlati megvalósításhoz elegendő, ha minden szinthez külön frekvencia táblát definiálunk, mivel a frekvencia táblákban szereplő frekvenciák hányadosai adják a próbákat (predikciókat).

Külön hangsúlyt érdemel a „binarizáció”.

Ez alatt azt értjük hogy 8-nál kisebb értékek esetén N-1 nullát írunk ki és egy egyest. Az N maga a szám értéke. Vagyis 7 esetén 6 nullát és egy-egyest írunk ki.

Binarizált	bináris érték	érték
1	00000001	1
01	00000010	2
001	00000011	3
0001	00000100	4
00001	00000101	5
000001	00000110	6
0000001	00000111	7

A táblázatból láthatjuk hogy a hagyományos bináris érték minden esetben 8 bit hosszúságú, még a „binarizált” változó hosszúságú, továbbá jobb entrópiát kapunk a „binarizálta” eredményül.

A „binarizációhoz” egy kontrol szintet is kell alkalmazni, ami megadja, hogy az adott byte „binarizált” vagy sem az adatfolyamban.

A 0. szint a kontrol, ahol 0-át kódolunk, ha az adat „binarizált” és 1-et, ha nem. Az egyes szinten kódoljuk a „binarizált” adatokat és a 2. szinten a normál adatokat.

Összegezve minden funkcionálisan elkülönülő adatstruktúra egy csoportot alkot, ami szintekre van bontva.

Tipikusan a mozgás vektorok, a koefficiensek, és egyéb adatok alkotják a csoportokat.

Végezetül elmondhatjuk, hogy a kódolás sebessége alig marad el a hagyományos videó technikában használt Huffman eljárástól, viszont annál sokkal hatékonyabb.

AMT AB

Szabadalmi igénypontok

1. Eljárás digitálisan kódolt, egy-egy időpillanathoz rendelt képek (frame) sorozatából álló mozgóképek tömörítésére, amely eljárás során bizonyos frame-ek információtartalmát az előtte vagy utána következő képek (referencia képek) tartalmából kódolunk (predikció), oly módon, hogy a referencia képekben előre meghatározott méretű blokkokon belül hasonló képeket keresünk, és a blokkokon belül az eredeti kép és a referencia képből kódolt adatokból ismét előállított kép közötti különbségeket megvizsgáljuk, és a különbségek alapján hibát számolunk, és ha a hiba nem halad meg egy előre meghatározott értéket, a kép tárolása helyett a referencia képből kódolt adatokat tároljuk,

azzal jellemezve, hogy

a kiszámolt hiba függvényében egy olyan blokkot, ami a hibához egy küszöbértéknél nagyobb mértékben járul hozzá, további részblokkokra osztunk, és a kódolás során a az eredeti kép és a kódolt adatokból helyreállított kép közötti különbségeket a részblokkok között számítjuk ki.

2. Az 1. igénypont szerinti eljárás, azzal jellemezve, hogy a kódolandó frame blokkjaihoz tartozó referencia blokkok keresése során a blokkokhoz hibatérképet készítünk, és a részblokkokra történő felbontást a hibatérkép alapján végezzük.

3. Az 1. igénypont szerinti eljárás, azzal jellemezve, hogy amennyiben egy részbloknál a hiba egy előre meghatározott értéknél nagyobb, a részblokkot további kisebb részblokkokra bontjuk, és a kisebb részblokkokkal ismét keresést végzünk az adott referencia frame-ben.

4. Az 1. igénypont szerinti eljárás, azzal jellemezve, hogy amennyiben egy vagy több részblokkban előforduló hiba miatt az eredeti blokkban a teljes hiba egy előre meghatározott értéknél nagyobb, a részblokkot tartalmazó nagyobb blokkal keresést végzünk további referencia frame-ekben (a hiba visszapropagálása).

5. Az 1-4. igénypontok bármelyike szerinti eljárás, azzal jellemezve, hogy 64x64, 32x32, 16x16, 8x8, 4x4 v. 2x2 méretű blokkokat és részblokkokat alkalmazunk.

6. Az 1-5. igénypontok bármelyike szerinti eljárás, azzal jellemezve, hogy külön kódoljuk a fényerő és szín adatokat (UV és Y blokkok).

7. Az 1-6. igénypontok bármelyike szerinti eljárás, azzal jellemezve, hogy továbbá ahol a digitálisan kódolt képek egyes képpontjaihoz rendelt információ mennyiségének csökkentését a szomszédos képpontokat leíró értékek átlagolásával vagy a digitális felbontás mélységének csökkentésével érjük el,

8.

Eljárás digitálisan kódolt, egy-egy időpillanathoz rendelt képek (frame) sorozatából álló mozgóképek tömörítésére, amely eljárás során bizonyos frame-ek információtartalmát az előtte vagy utána következő képek (referencia képek)

tartalmából kódolunk, oly módon, hogy a referencia képekben előre meghatározott méretű blokkokon belül hasonló képeket keresünk, és a blokkokon belül az eredeti kép és a referencia képből kódolt adatokból ismét előállított kép közötti különbségeket megvizsgáljuk, és a különbségek alapján hibát számolunk, és ha a hiba nem halad meg egy előre meghatározott értéket, a kép tárolása helyett a referencia képből kódolt adatokat tároljuk,

azzal jellemezve, hogy

amennyiben a kódolt adatok valószerű továbbításához szükséges sávszélesség egy adott küszöbértéket meghalad, a képméretet lecsökkentjük, és a kódolást a lecsökkentett képmérettel végezzük el.

a digitálisan kódolt képekhez rendelt információ mennyiségének csökkentését a kép méretének dinamikusa meghatározott mértékű csökkentésével érjük el, ahol a kép méretének meghatározását dinamikusan egy neurális hálózattal végezzük.

9. A 8. igénypont szerinti eljárás, azzal jellemezve, hogy a neurális hálózat bemenő adataként a frame típusát és/vagy a kódolandó kép paramétereit és/vagy a predikciós paramétereit használjuk.

10. Eljárás digitálisan kódolt, egy-egy időpillanathoz rendelt képek (frame) sorozatából álló mozgóképek tömörítésére, amely eljárás során bizonyos frame-ek információtartalmát az előtte vagy utána következő képek (referencia képek) tartalmából kódolunk, oly módon, hogy a referencia képekben előre meghatározott méretű blokkokon belül hasonló képeket keresünk, és a blokkokon belül az eredeti kép és a referencia képből kódolt adatokból ismét előállított kép közötti különbségeket megvizsgáljuk, és a különbségek alapján hibát számolunk, és ha a hiba nem halad meg egy előre meghatározott értéket, a kép tárolása helyett a referencia képből kódolt adatokat tároljuk,

azzal jellemezve, hogy

amennyiben egy blokkhoz nem találunk megfelelő referencia blokkot az előző vagy következő referencia frame-ban, az adott blokkhoz további referencia blokkot keresünk a többi referencia frame-ban.

11. igénypont szerinti eljárás, azzal jellemezve, hogy 2-5 frame távolságra keresünk további referencia frame-et.

12. Eljárás digitálisan kódolt, egy-egy időpillanathoz rendelt képek (frame) sorozatából álló mozgóképek tömörítésére, amely eljárás során bizonyos frame-ek információtartalmát az előtte vagy utána következő képek (referencia képek) tartalmából kódolunk, oly módon, hogy a referencia képekben előre meghatározott méretű blokkokon belül hasonló képeket keresünk, és a blokkokon belül az eredeti kép és a referencia képből kódolt adatokból ismét előállított kép közötti különbségeket megvizsgáljuk, és a különbségek alapján hibát számolunk, és ha a hiba nem halad meg egy előre meghatározott értéket, a kép tárolása helyett a referencia képből kódolt adatokat tároljuk,

azzal jellemezve, hogy

a képek kódolását különböző blokkméretekkel és különböző referenciák alapján végezzük el, majd az így elvégzett kódolásokat egymással összehasonlítva, a leghatékonyabb kódolást használjuk.

13. A 12. igénypont szerinti eljárás, azzal jellemezve, hogy a kódolás során aritmetikai kódolással tömörítjük a kódolt adatokat.

14. Eljárás digitálisan kódolt, egy-egy időpillanathoz rendelt képek (frame) sorozatából álló mozgóképek tömörítésére, amely eljárás során bizonyos frame-ek információtartalmát az előtte vagy utána következő képek (referencia képek) tartalmából kódolunk, oly módon, hogy a referencia képekben előre meghatározott méretű blokkokon belül hasonló képeket keresünk, és a blokkokon belül az eredeti kép és a referencia képből kódolt adatokból ismét előállított kép közötti különbségeket megvizsgáljuk, és a különbségek alapján hibát számolunk, és ha a hiba nem halad meg egy előre meghatározott értéket, a kép tárolása helyett a referencia képből kódolt adatokat tároljuk,

azzal jellemezve, hogy

a digitálisan kódolt képekhez rendelt információ mennyiségének csökkentését a kép méretének dinamikusan meghatározott mértékű csökkentésével érjük el.

15. A 14. igénypont szerinti eljárás, azzal jellemezve, hogy a kép méretének meghatározását dinamikusan egy neurális hálózattal végezzük.

16. A 15. igénypont szerinti eljárás, azzal jellemezve, hogy a neurális hálózat bemenő adataként a rendelkezésre álló sáv szélességet és/vagy a kódolás hibáját és/vagy a frame típusát és/vagy a kódolandó kép paramétereit és/vagy a predikciós paramétereit használjuk.

17. Eljárás digitálisan kódolt, egy-egy időpillanathoz rendelt képek (frame) sorozatából álló mozgóképek tömörítésére, amely eljárás során bizonyos frame-ek információtartalmát az előtte vagy utána következő képek (referencia képek) tartalmából kódolunk, oly módon, hogy a referencia képekben előre meghatározott méretű blokkokon belül hasonló képeket keresünk, és a blokkokon belül az eredeti kép és a referencia képből kódolt adatokból ismét előállított kép közötti különbségeket megvizsgáljuk, és a különbségek alapján hibát számolunk, és ha a hiba nem halad meg egy előre meghatározott értéket, a kép tárolása helyett a referencia képből kódolt adatokat tároljuk, továbbá ahol a kódolt adatokat aritmetikai kódolással tömörítjük,

azzal jellemezve, hogy

az aritmetikai kódolás paramétereit egy adaptív kódolóval számítjuk ki, ahol az adaptív kódoló a kódolandó bitfolyamban folyamatosan vizsgálja a karakterek eloszlását, és az aritmetikai kódolás valószínűségi paraméterét a megfigyelt eloszlás alapján képezi.

18. A 17. igénypont szerinti eljárás, azzal jellemezve, hogy az adaptív kódolóként egy léptető bitregisztert alkalmazunk, és az aritmetikai kódolás valószínűségi paramétereit a bitregiszterben pillanatnyilag tárolt bitek eloszlása alapján dinamikusan képezzük.

19. A 17. igénypont szerinti eljárás; azzal jellemezve, hogy a valószínűségi paramétert egy frekvencia tábla segítségével képezzük, aminek az értékeit a bitregiszter alapján frissítjük.

20. Berendezés az 1-19. igénypontok bármelyike szerinti eljárás végrehajtására.

21. Software, amely az 1-19. igénypontok bármelyike szerinti eljárást végrehajtó utasításokat tartalmaz.

22. Kódolt jelsorozat, ami az 1-19. igénypontok bármelyike szerinti tömörítő eljárás eredményeképpen állt elő.

23. Eljárás az mozgóképet tartalmazó adatoknak 1-19. igénypontok bármelyike szerinti tömörítő eljárás által előállított kódolt jelsorozatból történő kitömörítésre.

A meghatalmazott

KIVONAT

Eljárás és berendezés mozgókép adatok tömörítésére AMT Advanced Multimedia Technology AB, Karlshamn, SE

A találmány tárgya eljárás digitálisan kódolt, egy-egy időpillanathoz rendelt képek (frame) (16,17,18) sorozatából álló mozgóképek tömörítésére, amely eljárás során bizonyos frame-ek (16,17,18) információtartalmát az előtte vagy utána következő képek (referencia képek) tartalmából kódolunk (predikció). A kódolás során a referencia képekben előre meghatározott méretű blokkokon (21,19) belül hasonló képeket keresünk, és a blokkokon (21,19) belül az eredeti kép (17) és a referencia képből (16,18) kódolt adatokból ismét előállított kép közötti különbségeket megvizsgáljuk. A különbségek alapján hibát számolunk, és ha a hiba nem halad meg egy előre meghatározott értéket, a kép (17) tárolása helyett a referencia képből (16,18) kódolt adatokat tároljuk. A találmány értelmében úgy járunk el, hogy a kiszámolt hiba függvényében egy olyan blokkot (20), ami a hibához egy küszöbértéknél nagyobb mértékben járul hozzá, további részblokkokra osztunk, és a kódolás során a az eredeti kép (17) és a kódolt adatokból helyreállított kép közötti különbségeket a részblokkok között számítjuk ki

A találmány tárgya még olyan, hasonló elvű tömörítő eljárás, amelynél ha a kódolt adatok valósidejű továbbításához szükséges sávszélesség egy adott küszöbértéket meghalad, a képméretet lecsökkentjük, és a kódolást a lecsökkentett képmérettel végezzük el.

A találmány tárgya még olyan, hasonló elvű tömörítő eljárás, amelynél a digitálisan kódolt képekhez rendelt információ mennyiségének csökkentését a kép (17) méretének dinamikusan meghatározott mértékű csökkentésével érjük el, ahol a kép (17) méretének meghatározását dinamikusan egy neurális hálózattal végezzük.

A találmány tárgya továbbá olyan, hasonló elvű tömörítő eljárás, amelynél ha egy blokkhoz (20) nem találunk megfelelő referencia blokkot (19,21) az előző vagy következő referencia frame-ban, az adott blokkhoz (20) további referencia blokkot keresünk a többi referencia frame-ban.

A találmány tárgya még olyan, hasonló elvű tömörítő eljárás, amelynél a képek (16,17,18) kódolását különböző blokkméretekkel és különböző referenciák alapján végezzük el, majd az így elvégzett kódolásokat egymással összehasonlítva, a leghatékonyabb kódolást használjuk.

A találmány tárgya még olyan, hasonló elvű tömörítő eljárás, amelynél a digitálisan kódolt képekhez (16,17,18) rendelt információ mennyiségének csökkentését a kép (16,17,18) méretének dinamikusan meghatározott mértékű csökkentésével érjük el.

Jellemző ábra: 2. ábra

[illegible]

D.

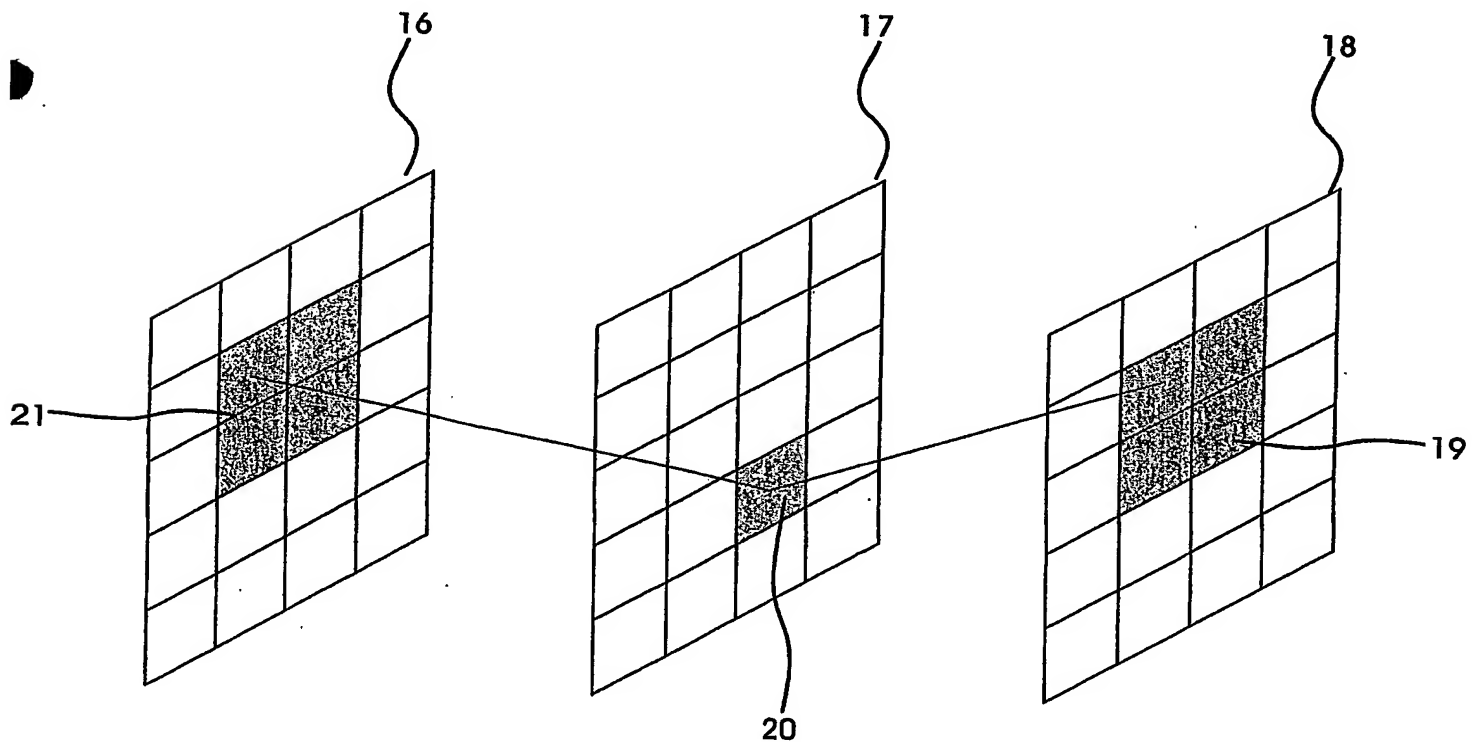


Fig 3

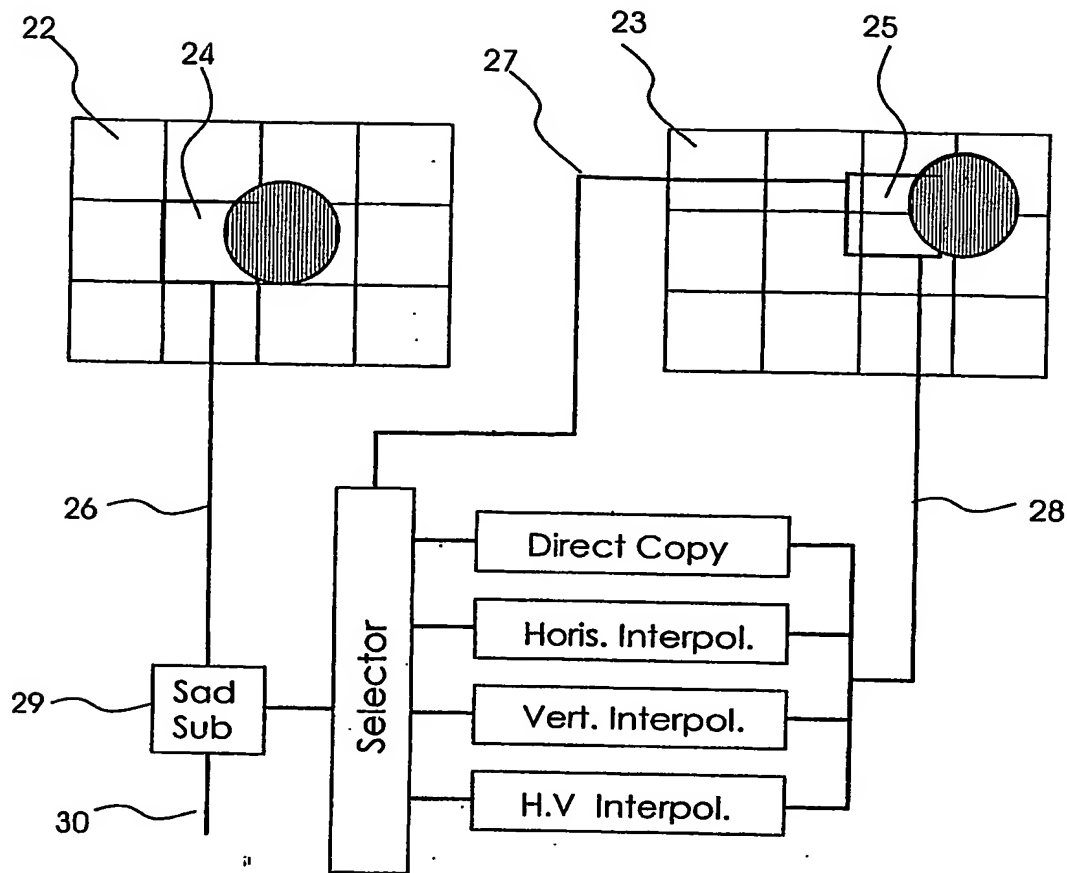


Fig 4

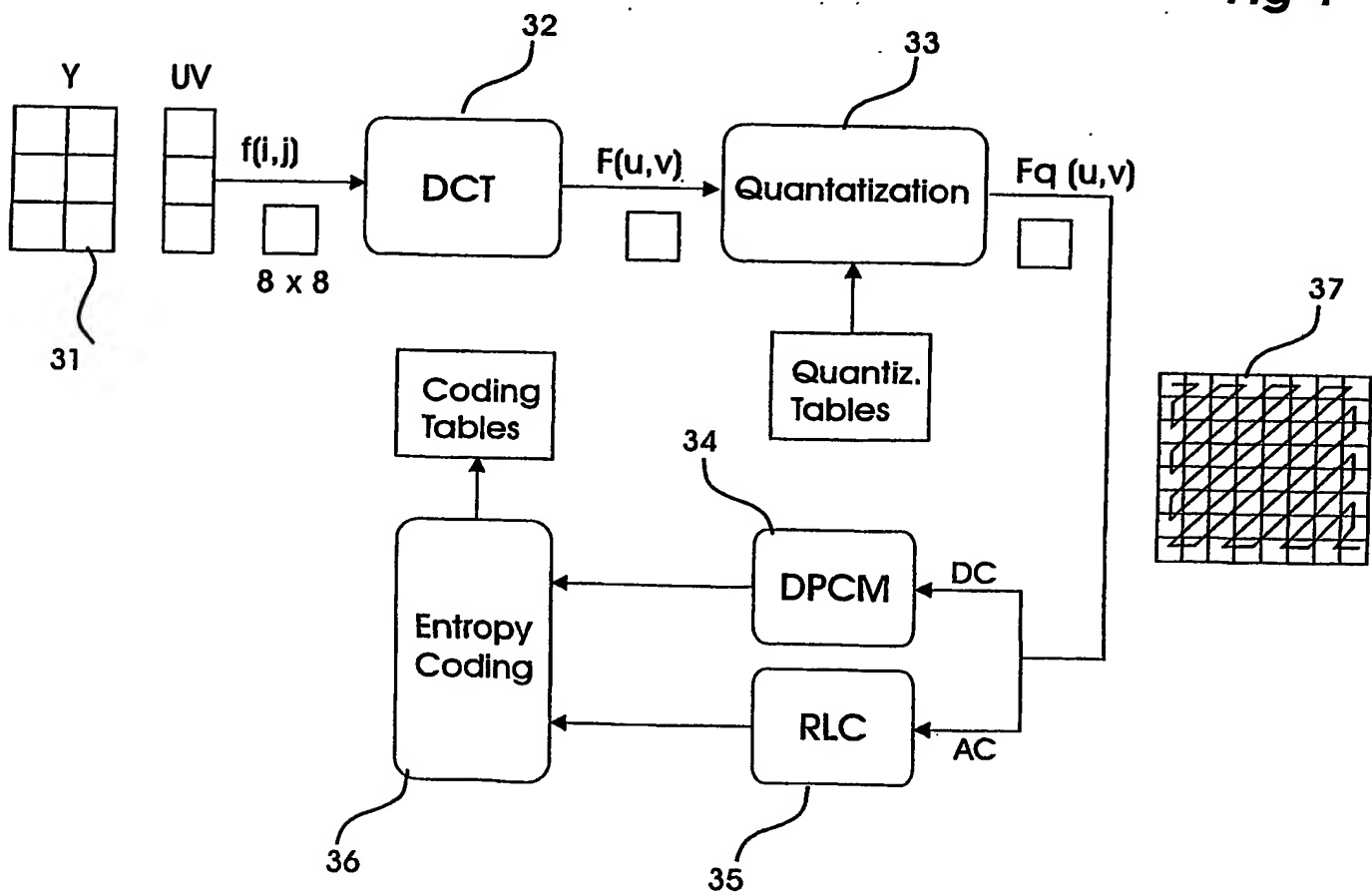


Fig 5

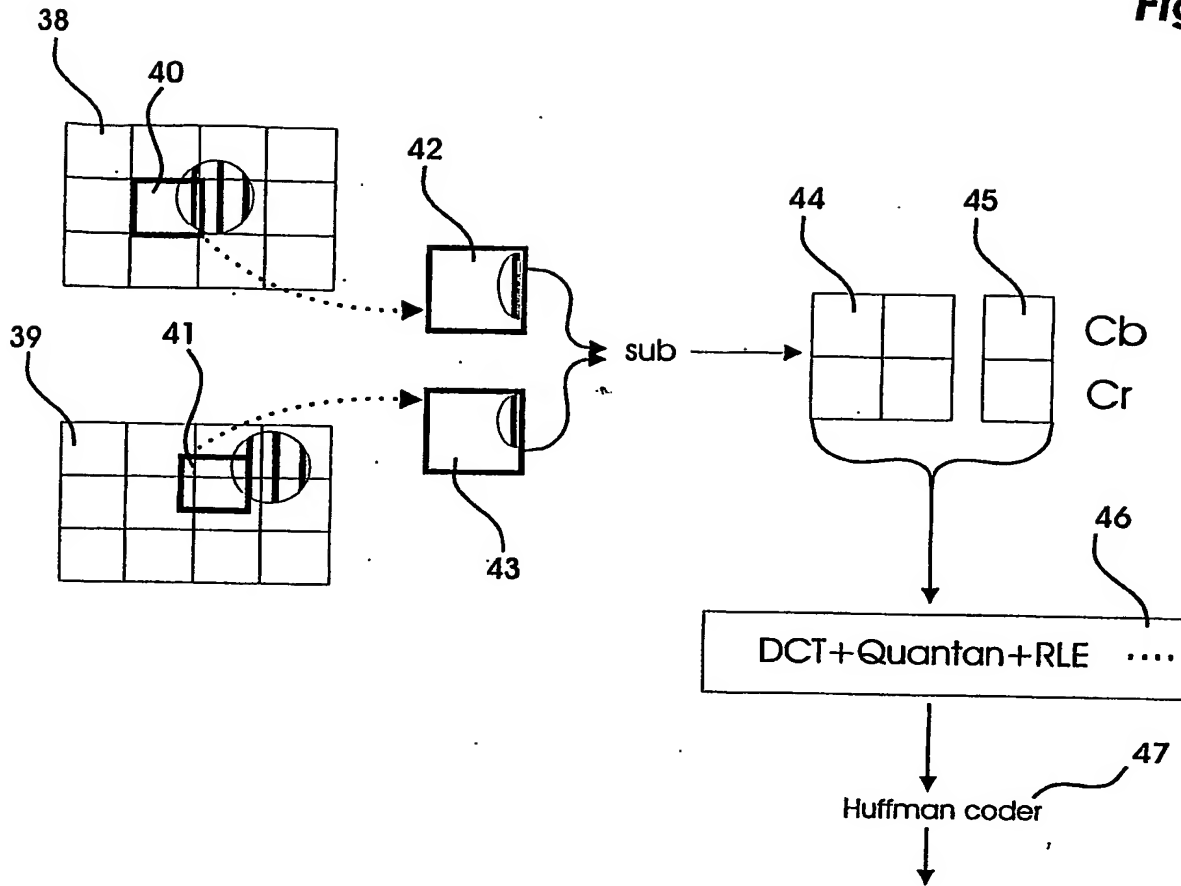


Fig 6

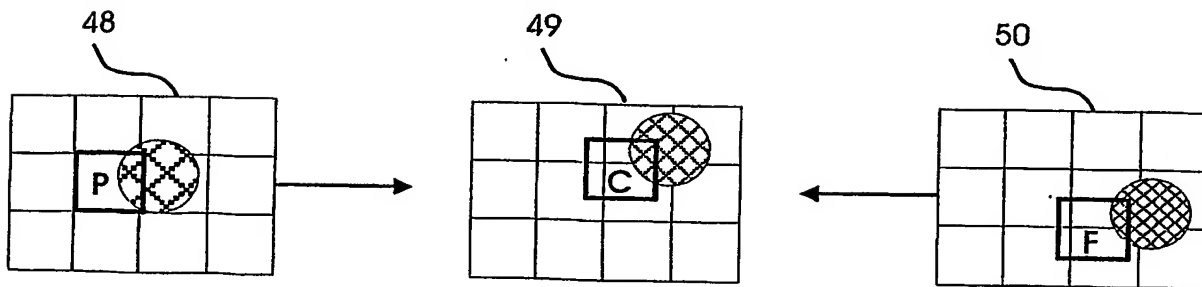


Fig 7

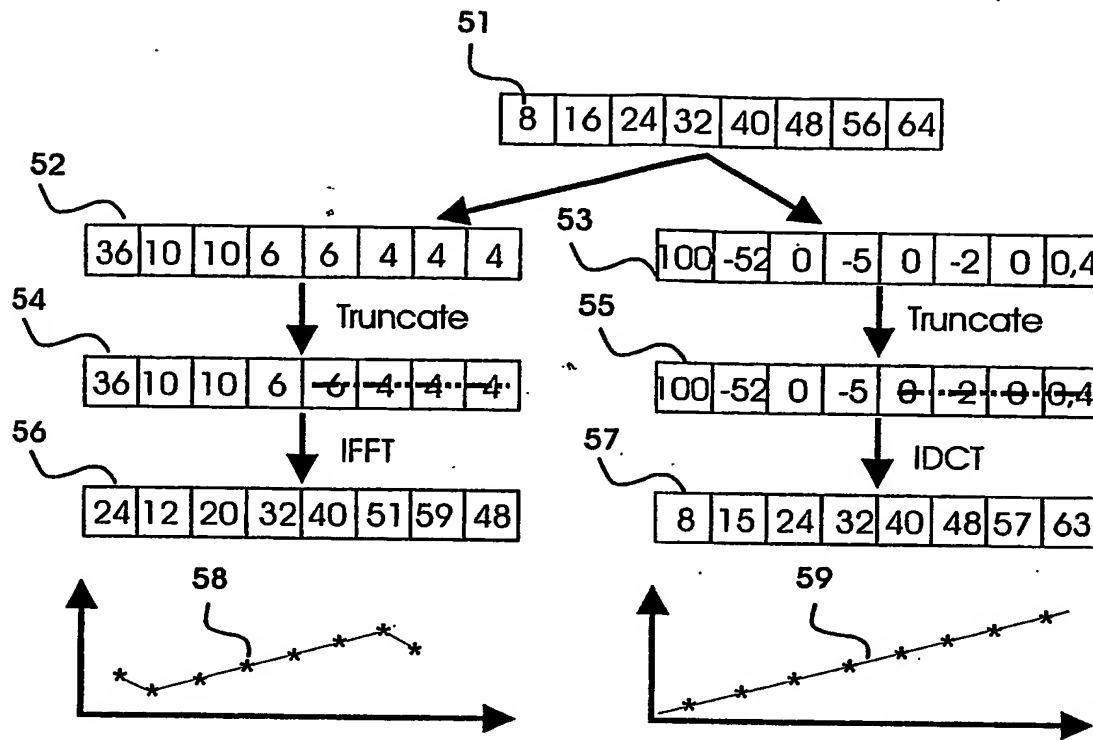


Fig 8

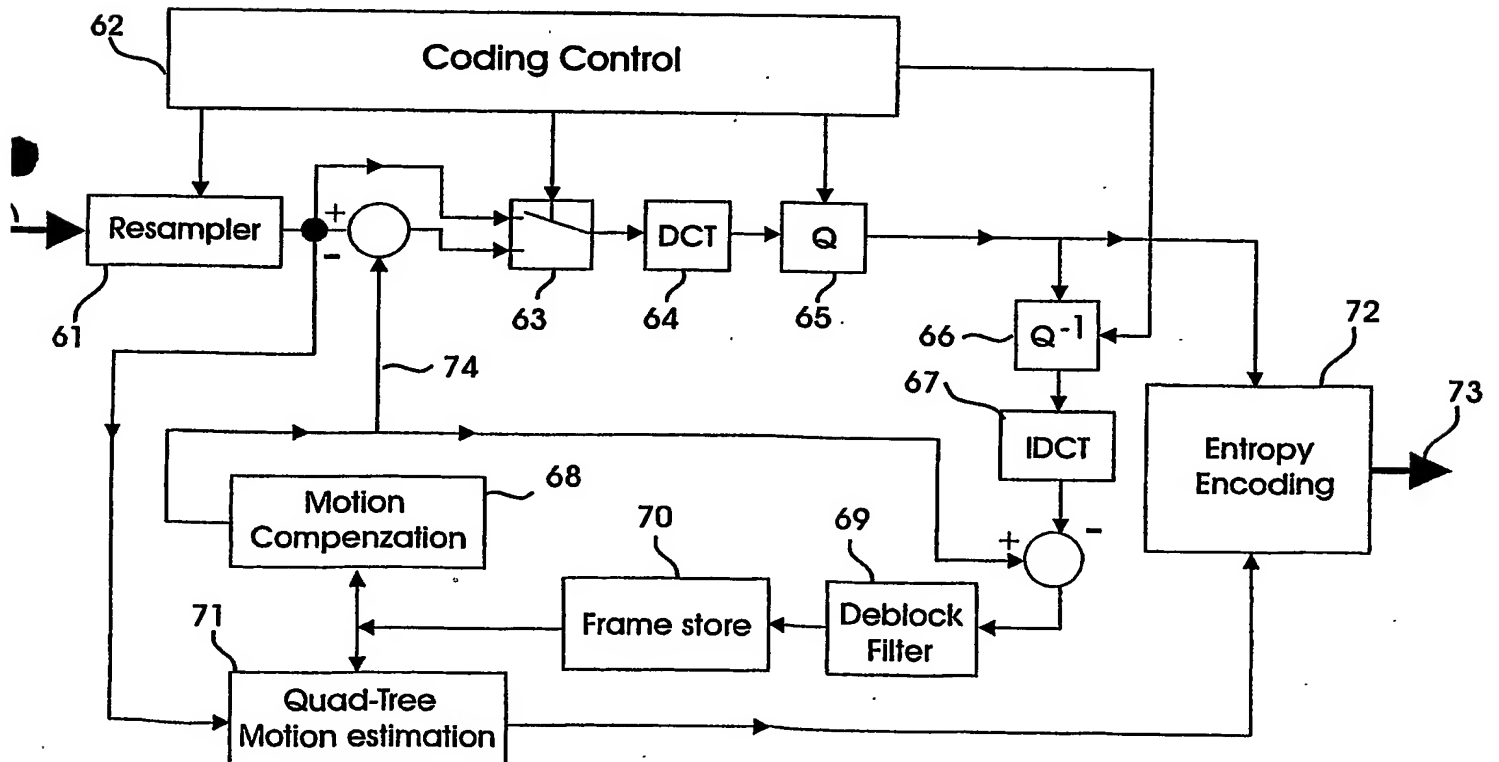


Fig 9

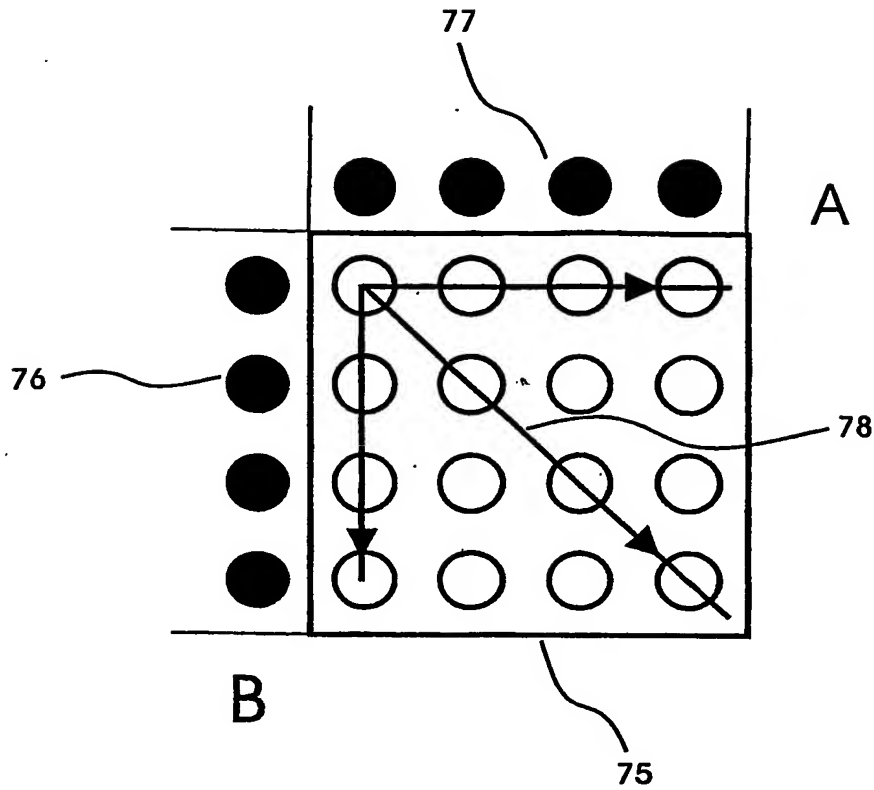
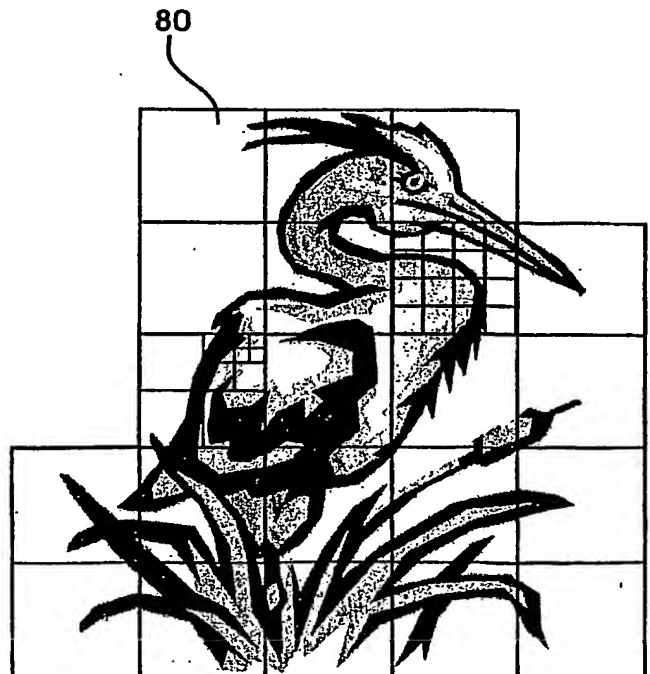
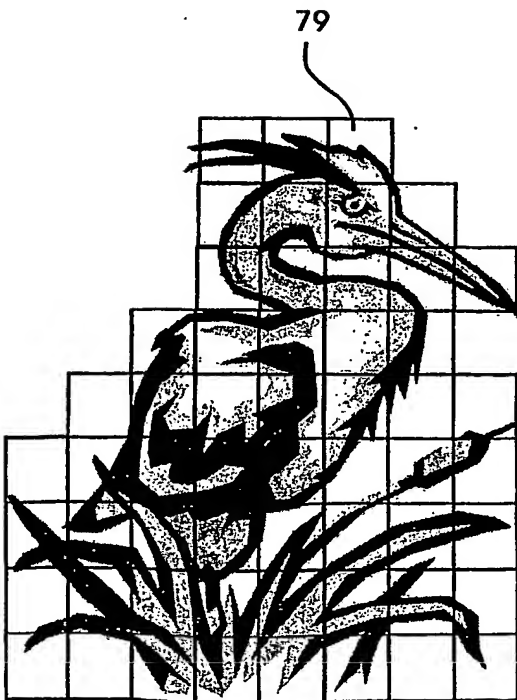


Fig 10



82

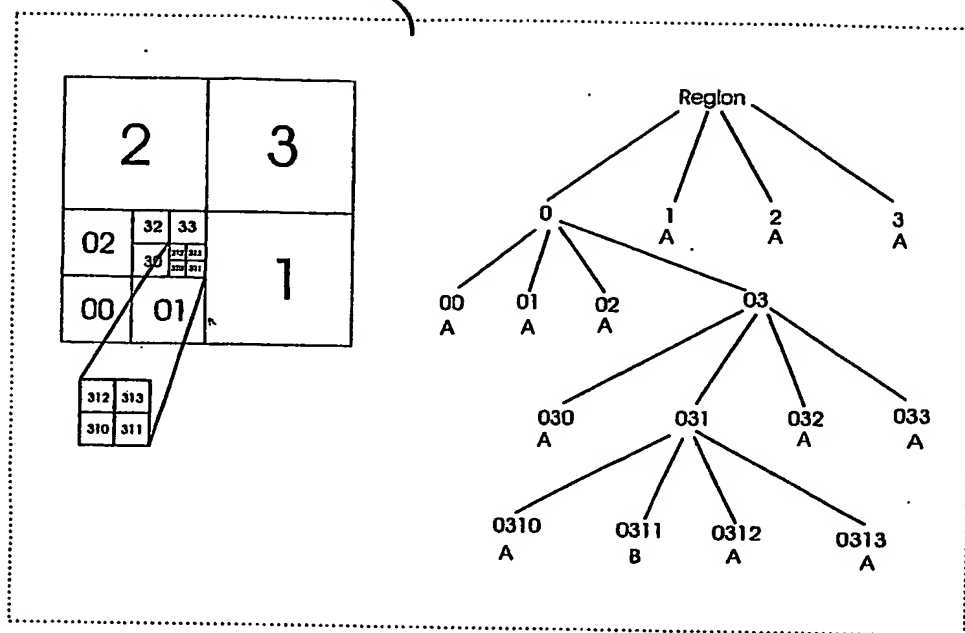
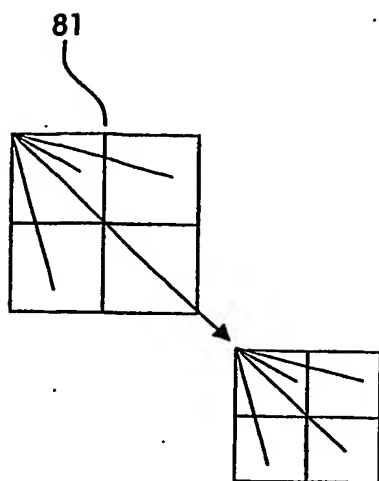


Fig 12

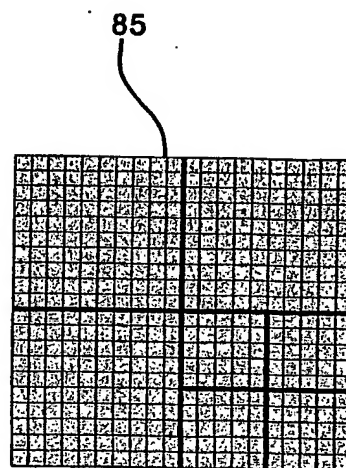
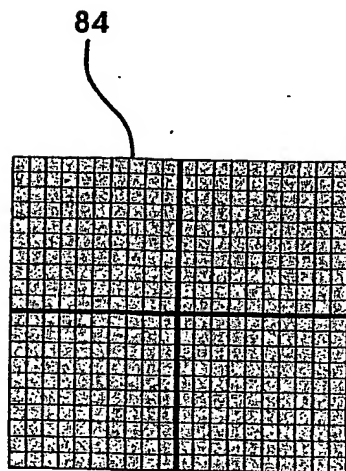
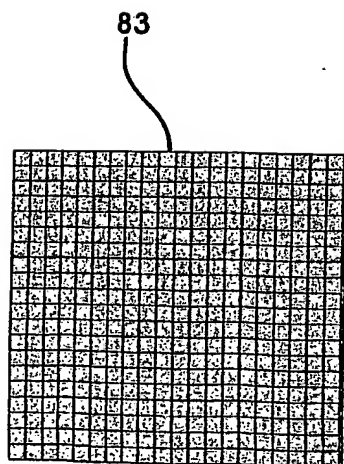


Fig 13a

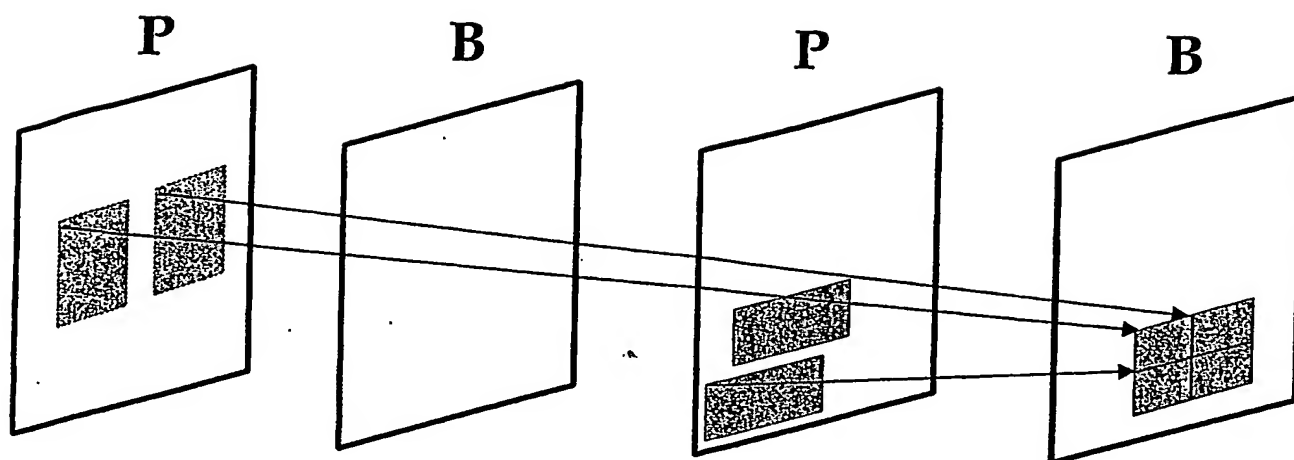


Fig 13b

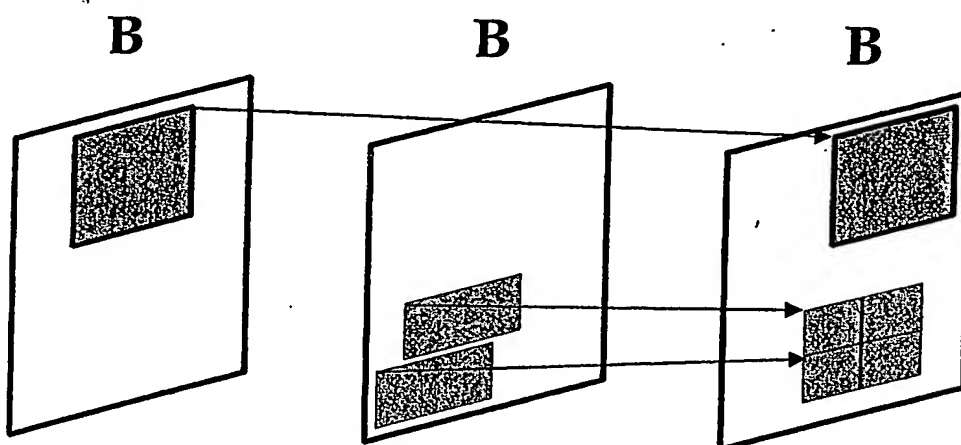


Fig 13c

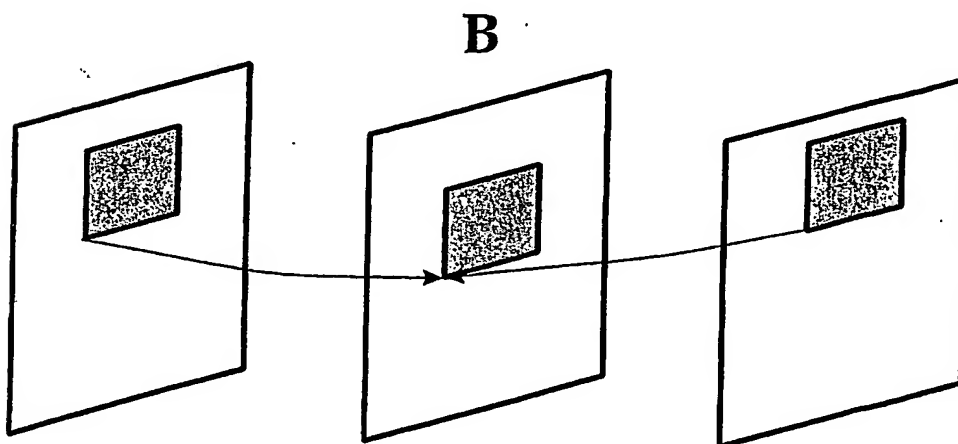


Fig 14 a

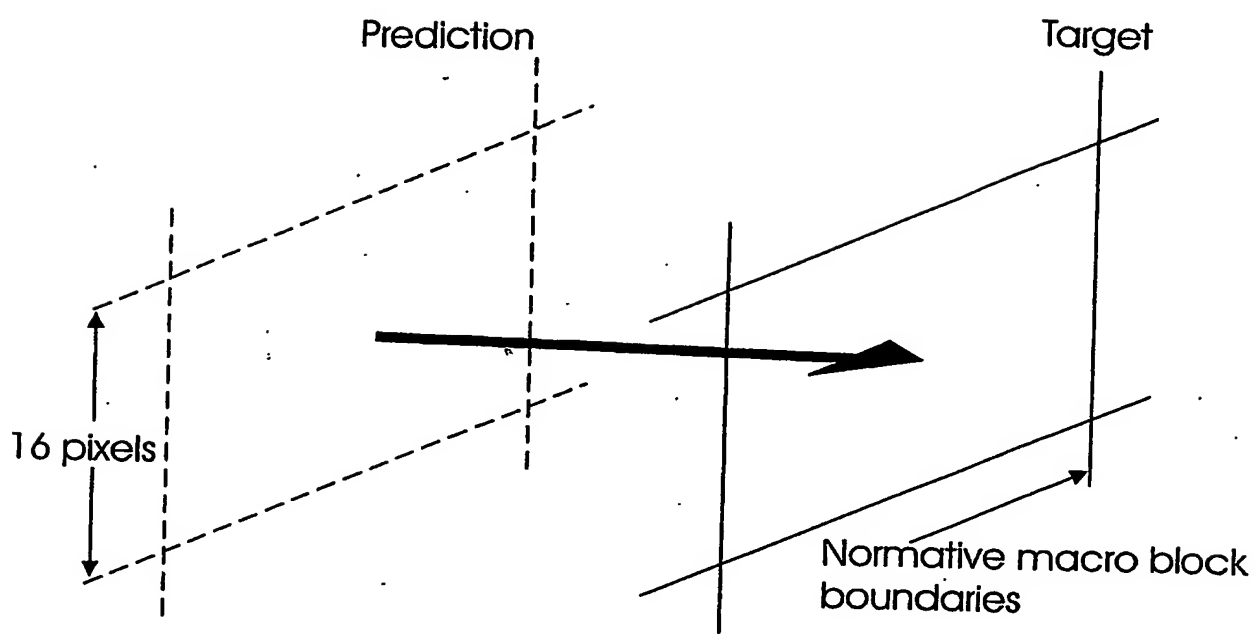


Fig 14 b

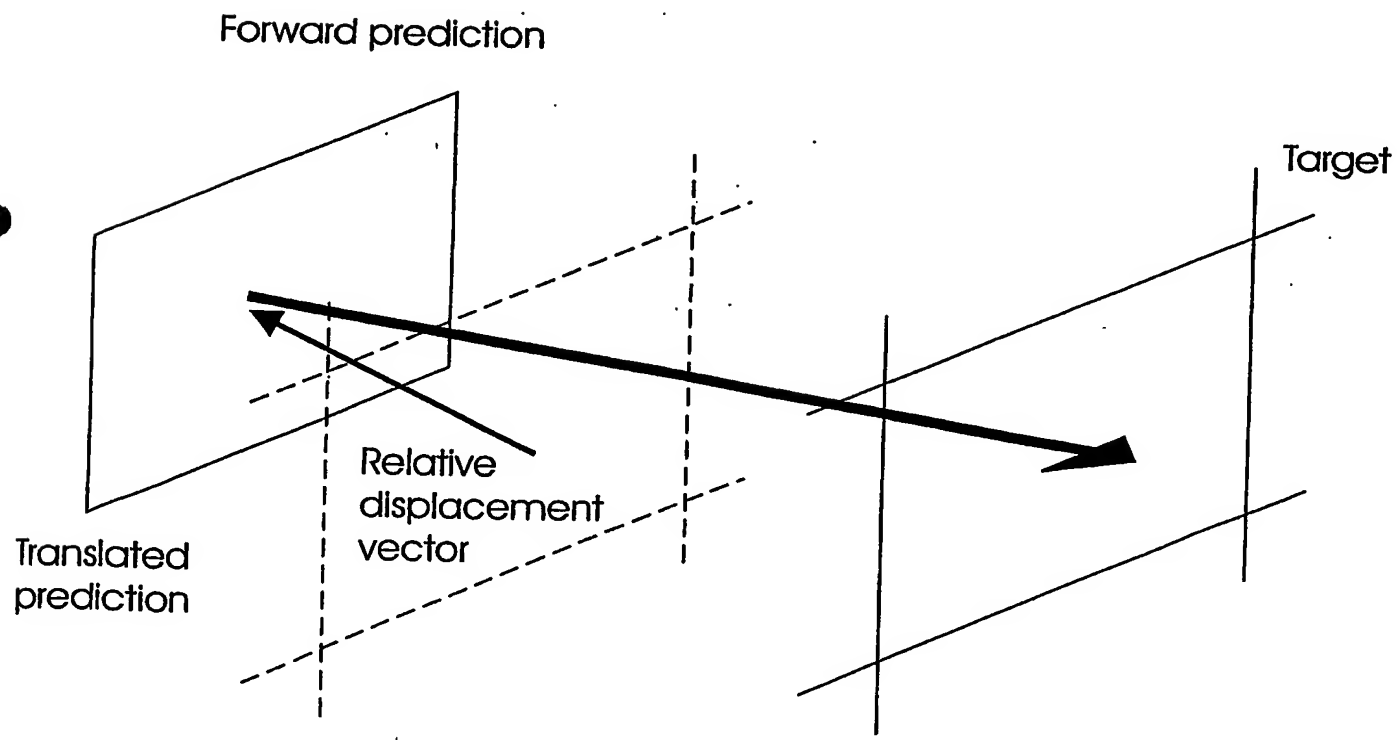


Fig 15

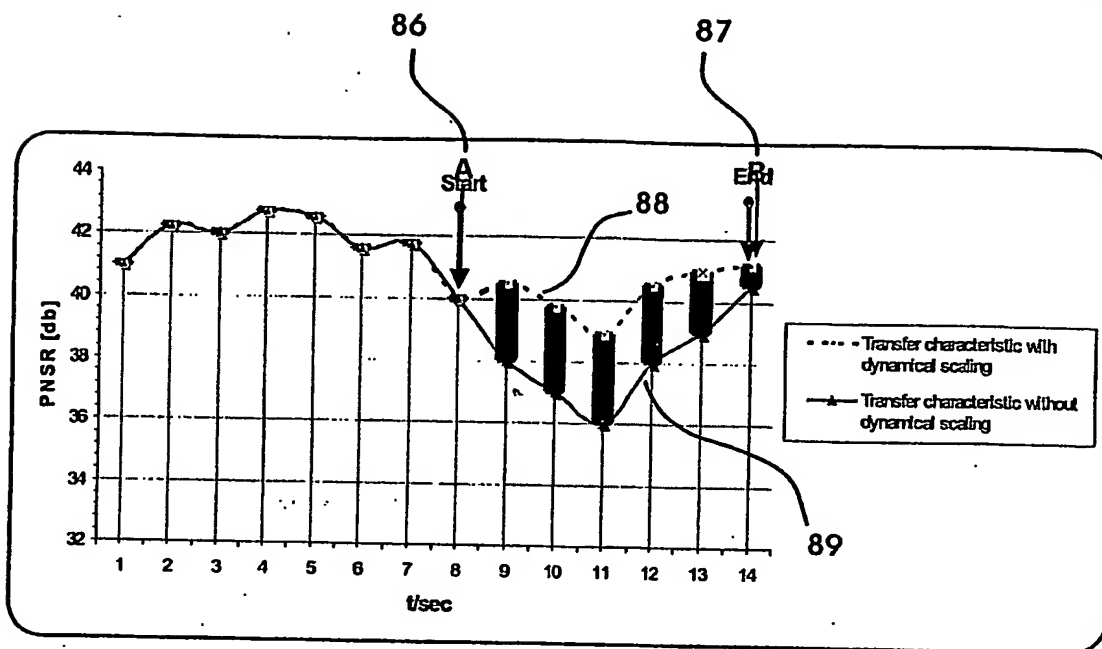


Fig 16

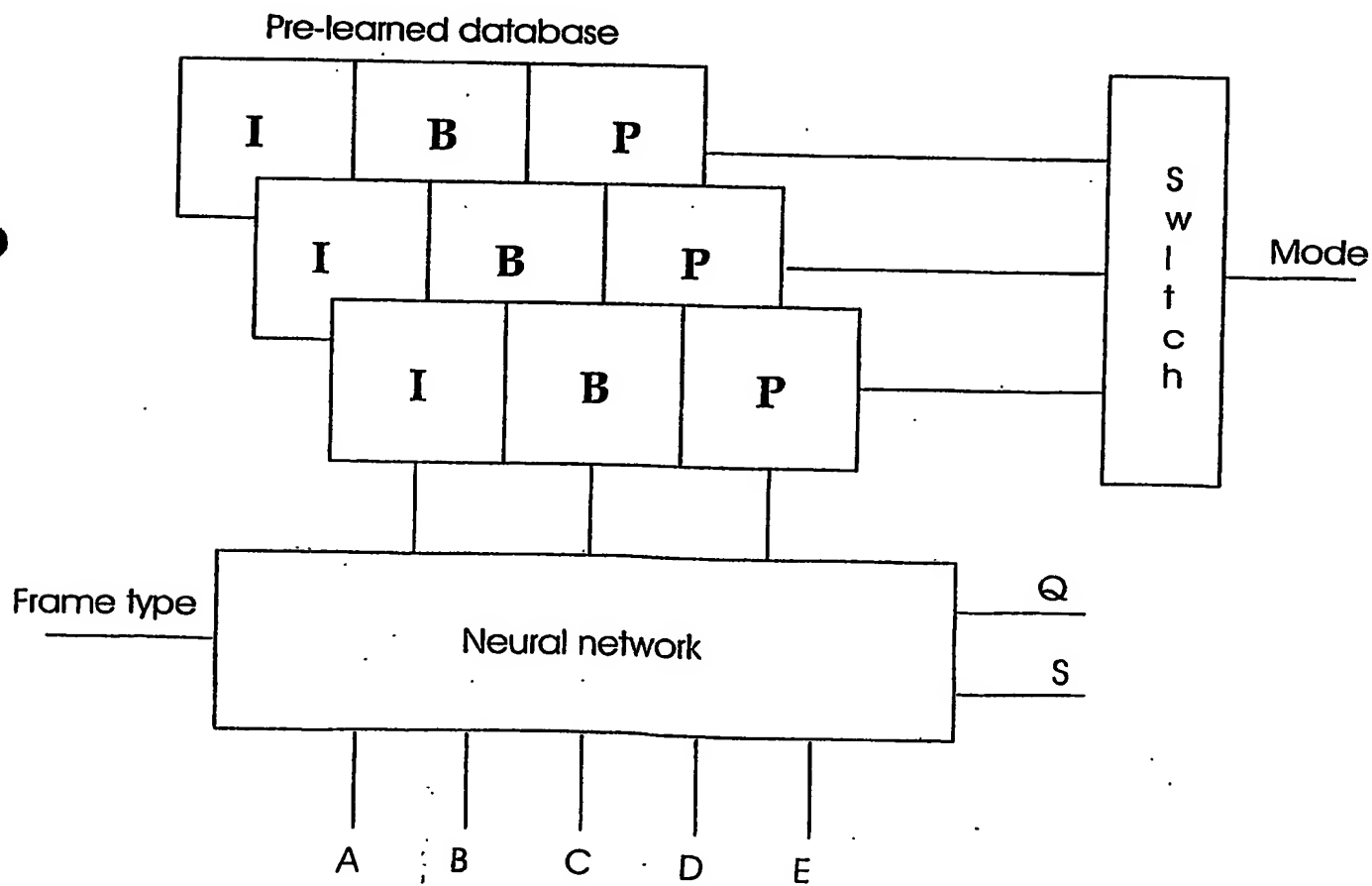


Fig 17

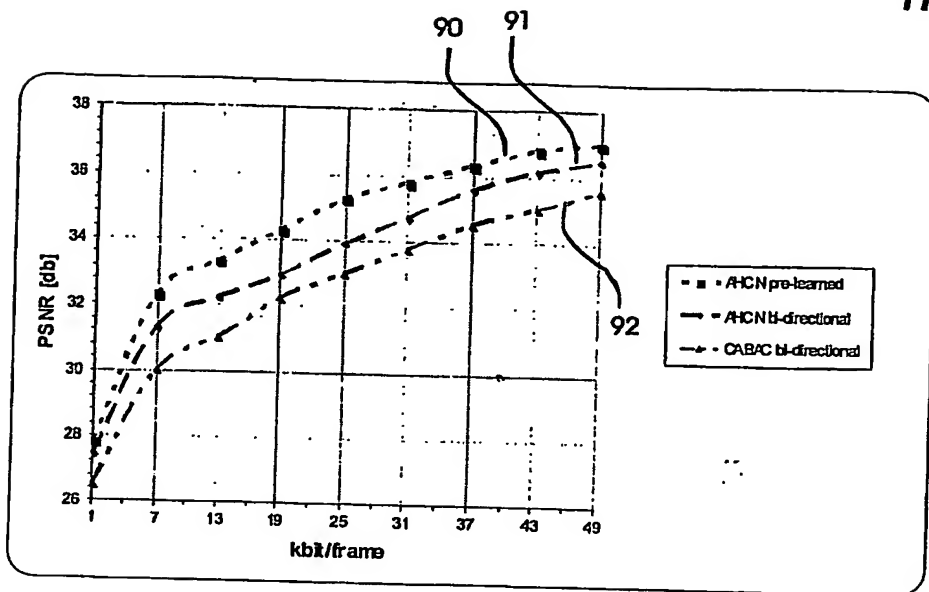
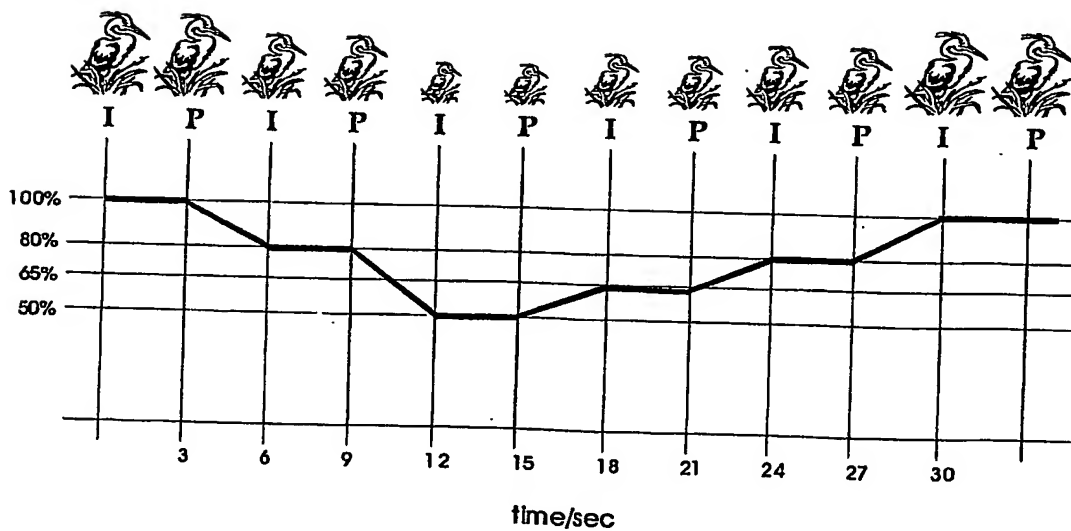


Fig 18



11/25

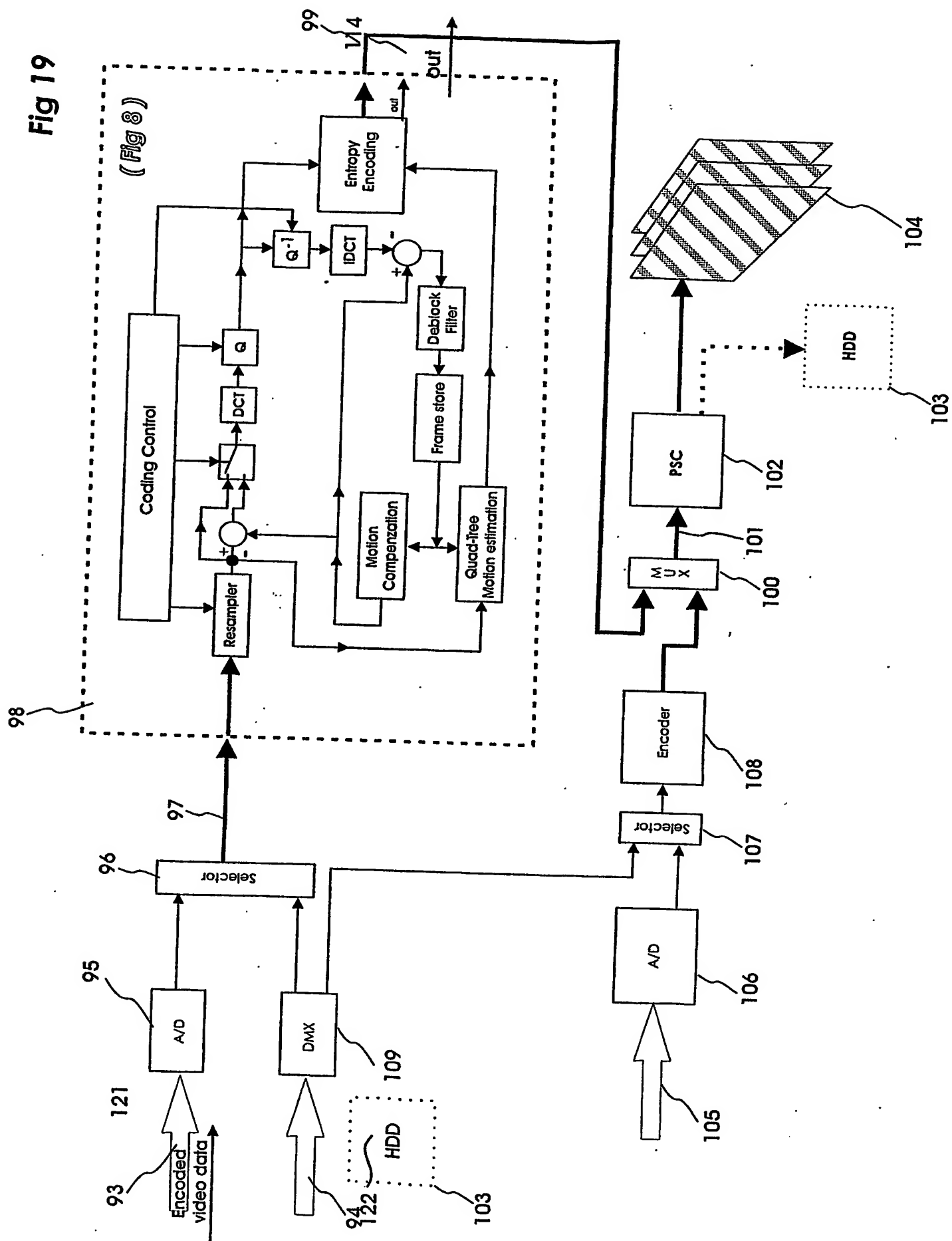


Fig 20

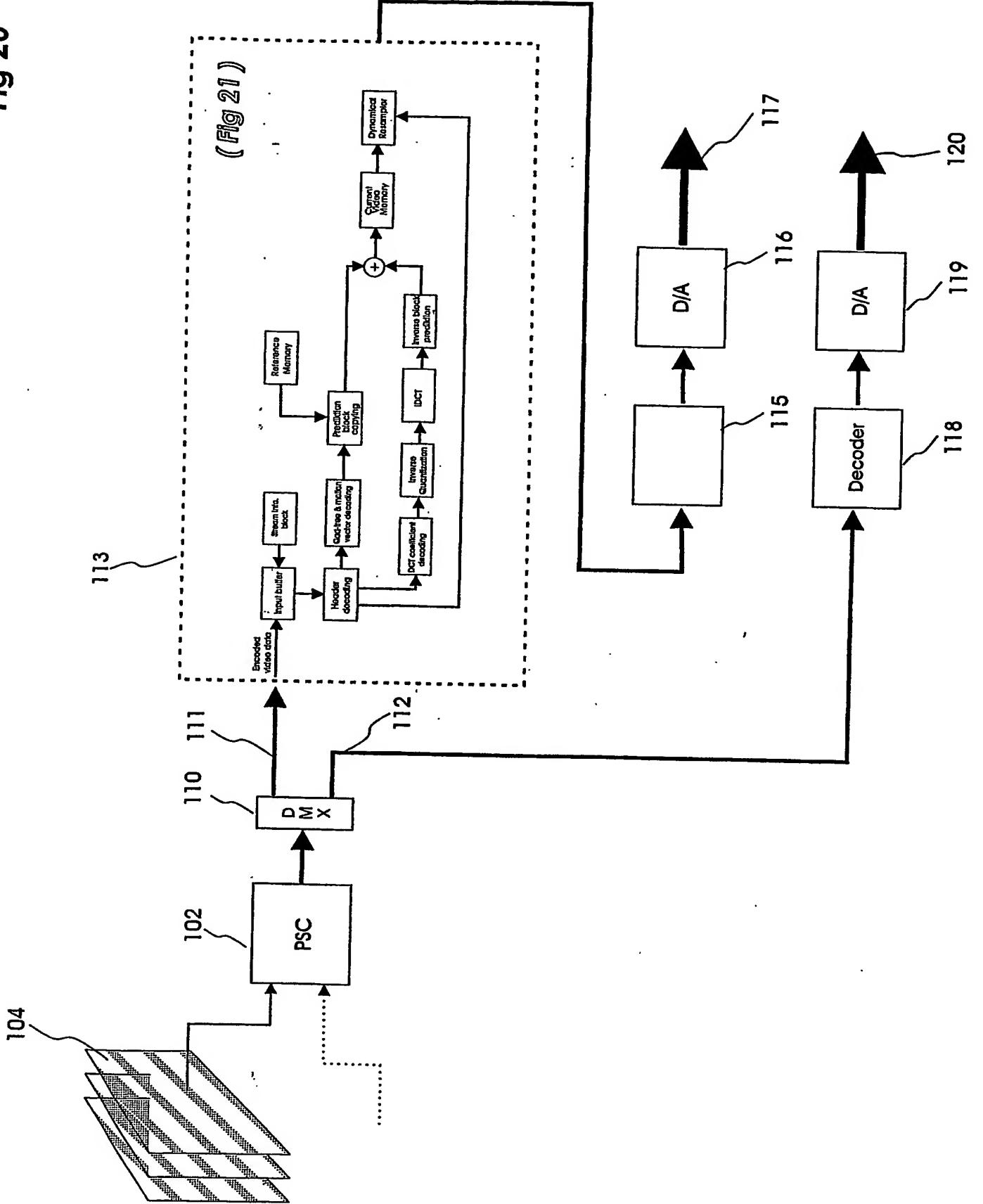
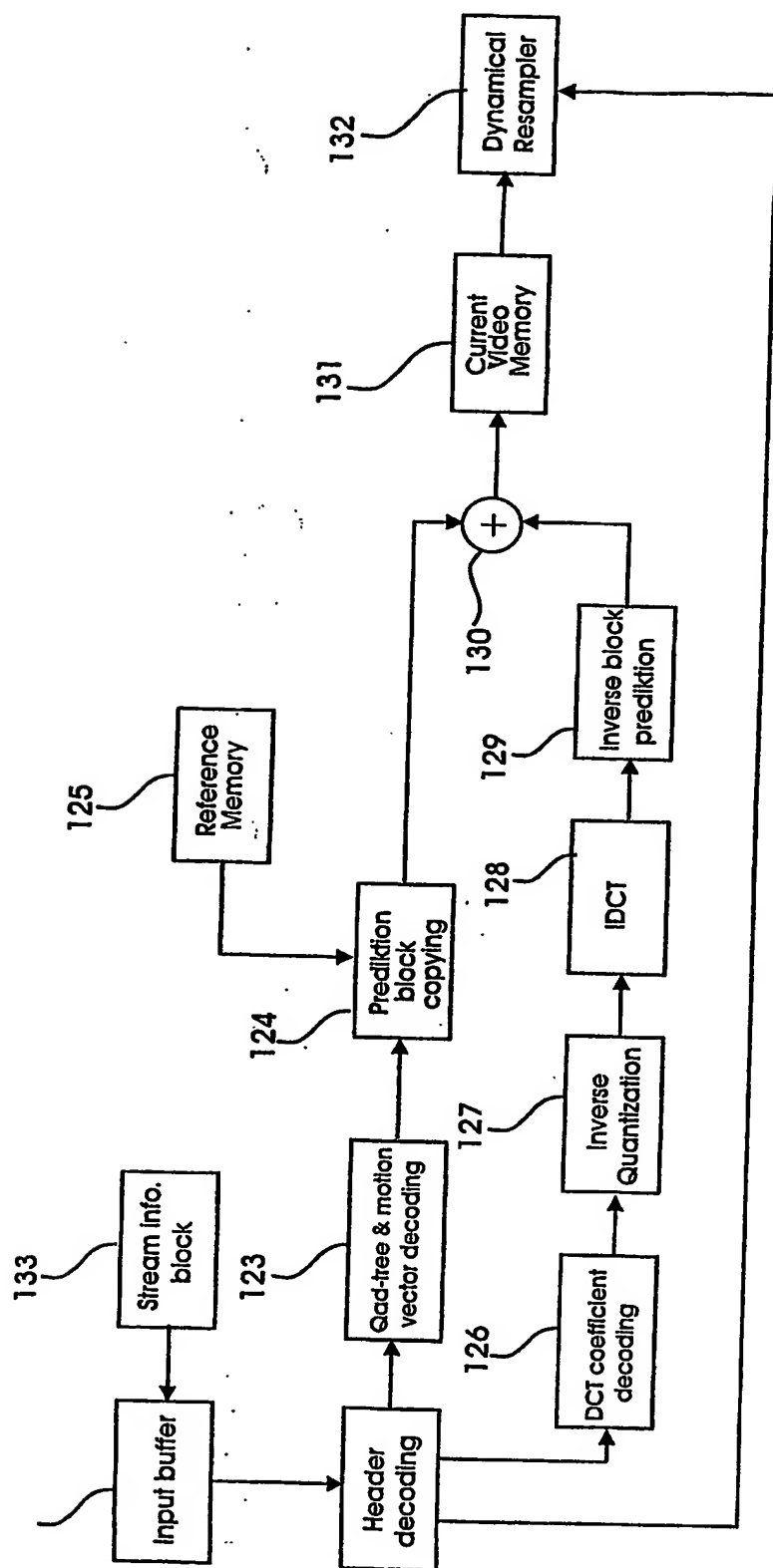


Fig 21



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.